

COMP0271: Inteligência Artificial

Busca com informação

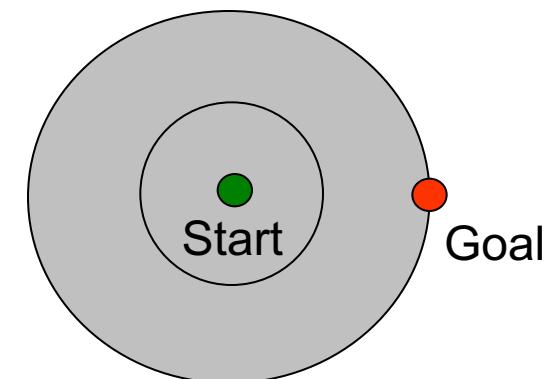
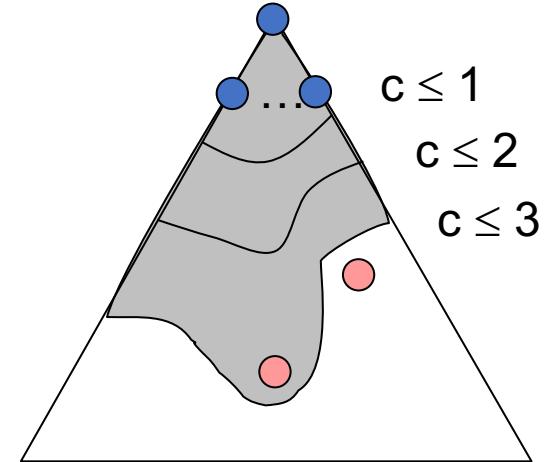


Professor: Hendrik Macedo

Universidade Federal de Sergipe, Brasil

Uniform Cost : observações

- UCS explora contornos de custos crescentes
- UCS é completo e ótimo (BOM)
- UCS explora opções em todas as direções (**não há dica** sobre local do estado objetivo) (RUIM)



Hora do último
contato?

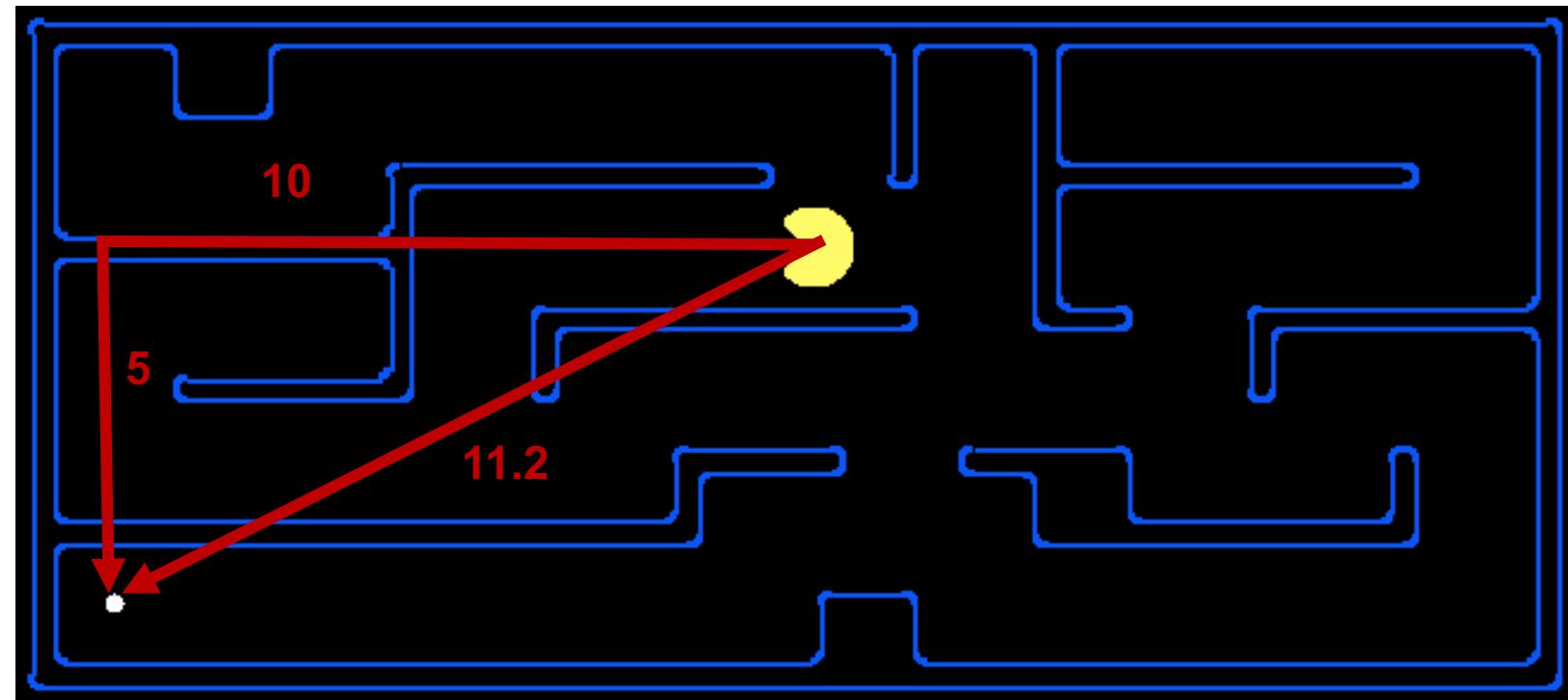
Sentido e força
do vento?

Correntes
marinhas?

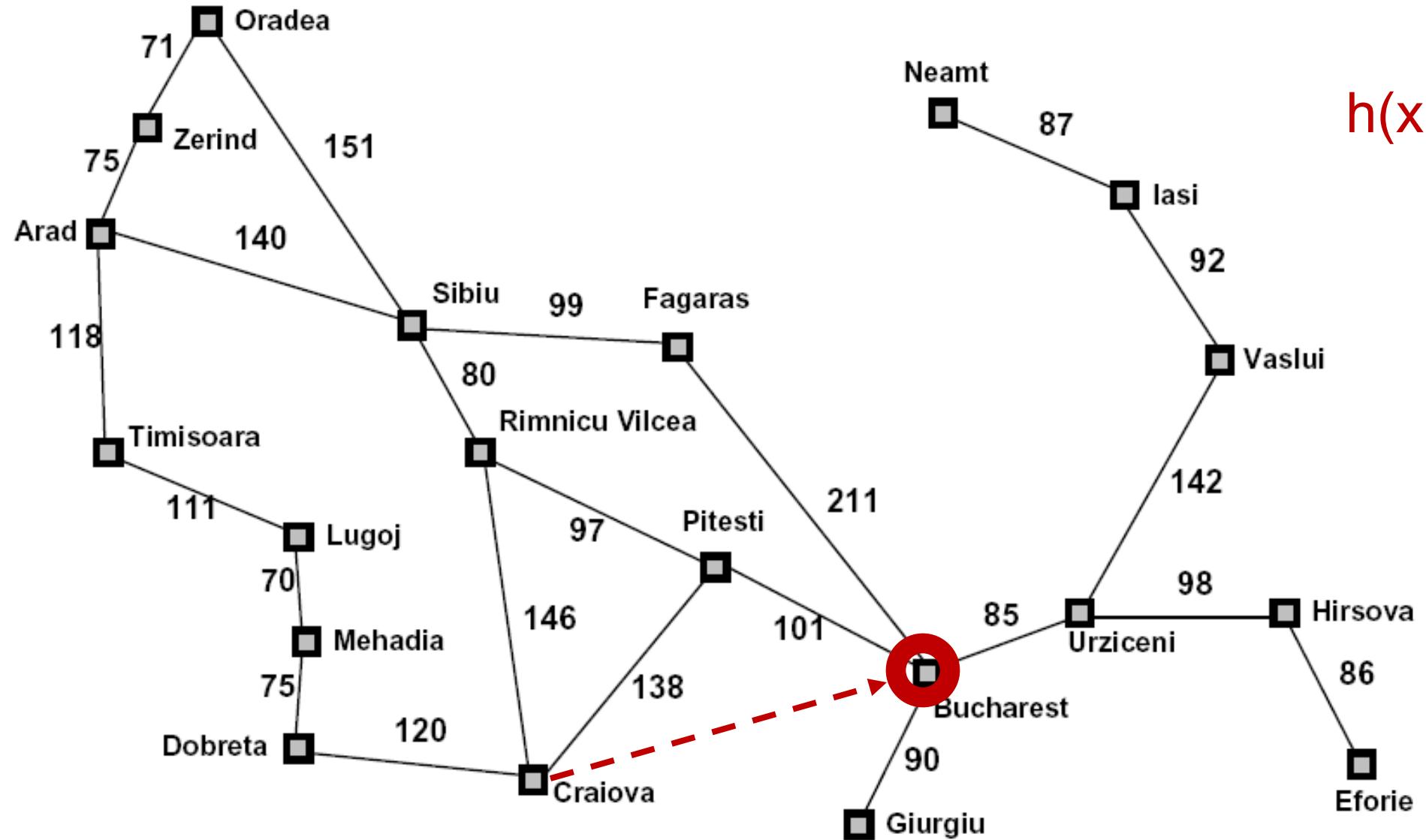


Heurística $h(x)$

- Função que **estima** o quanto um dado estado está perto do estado objetivo
- Criada/definida para cada problema em particular
- Exemplos: distância de Manhattan e distância Euclidiana em problemas de *pathfinding*

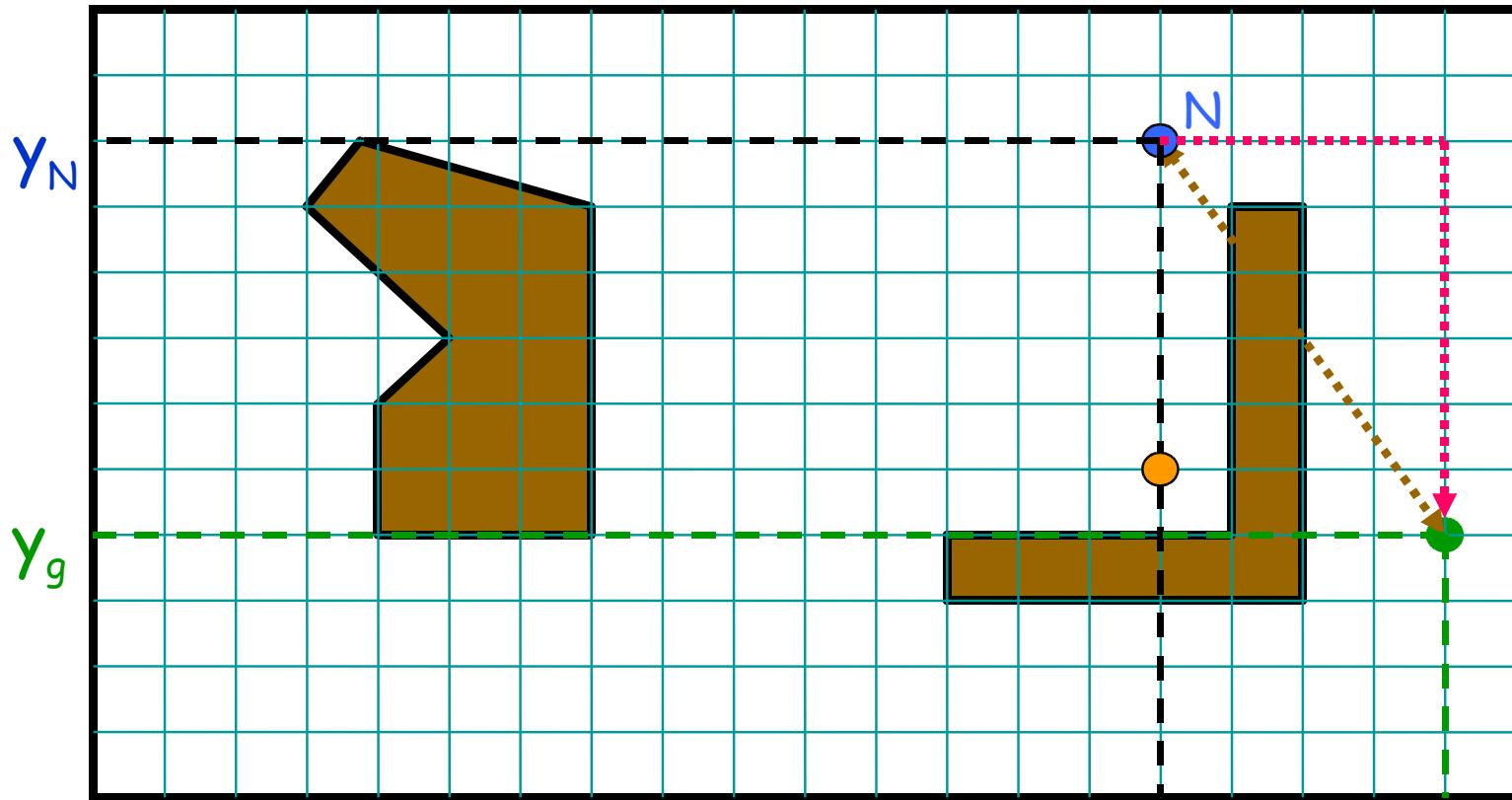


Exemplo



$h(x)$

Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



$$h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$$

$$h_2(N) = |x_N - x_g| + |y_N - y_g|$$

x_N x_g
(distância euclidiana)

(distância de Manhattan)

Greedy Search (busca gulosa)

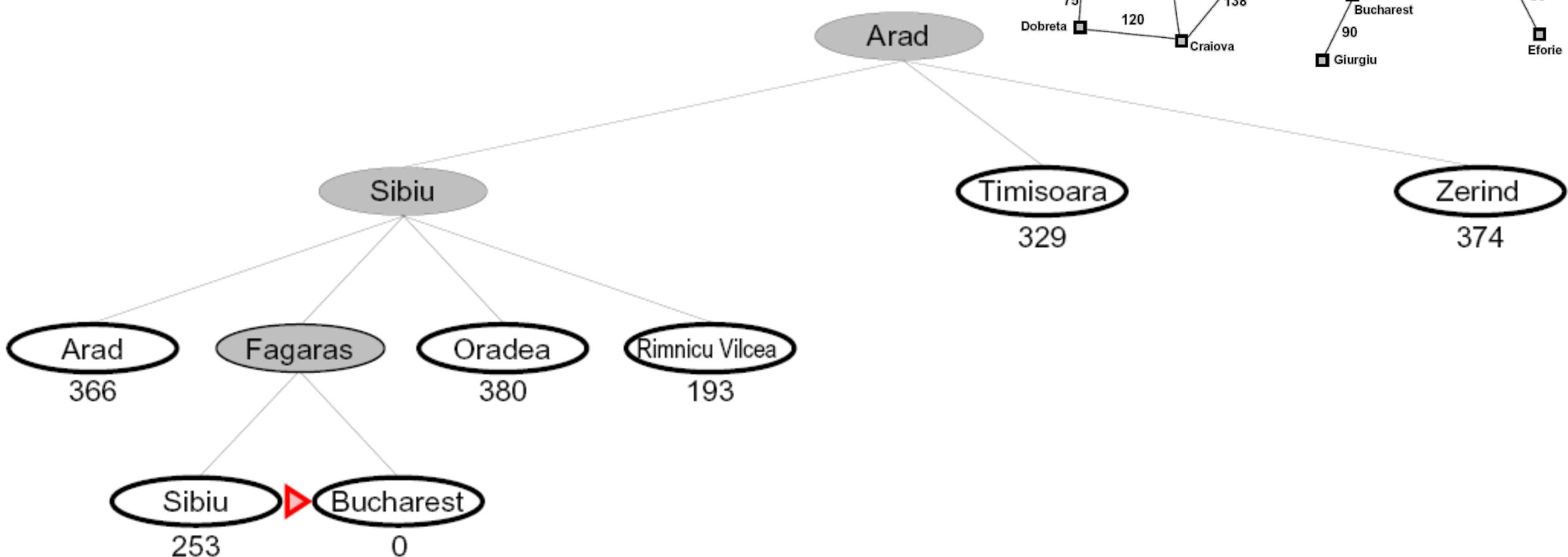


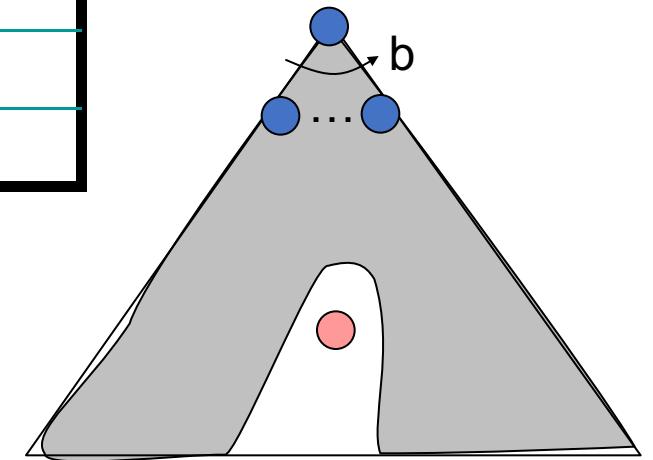
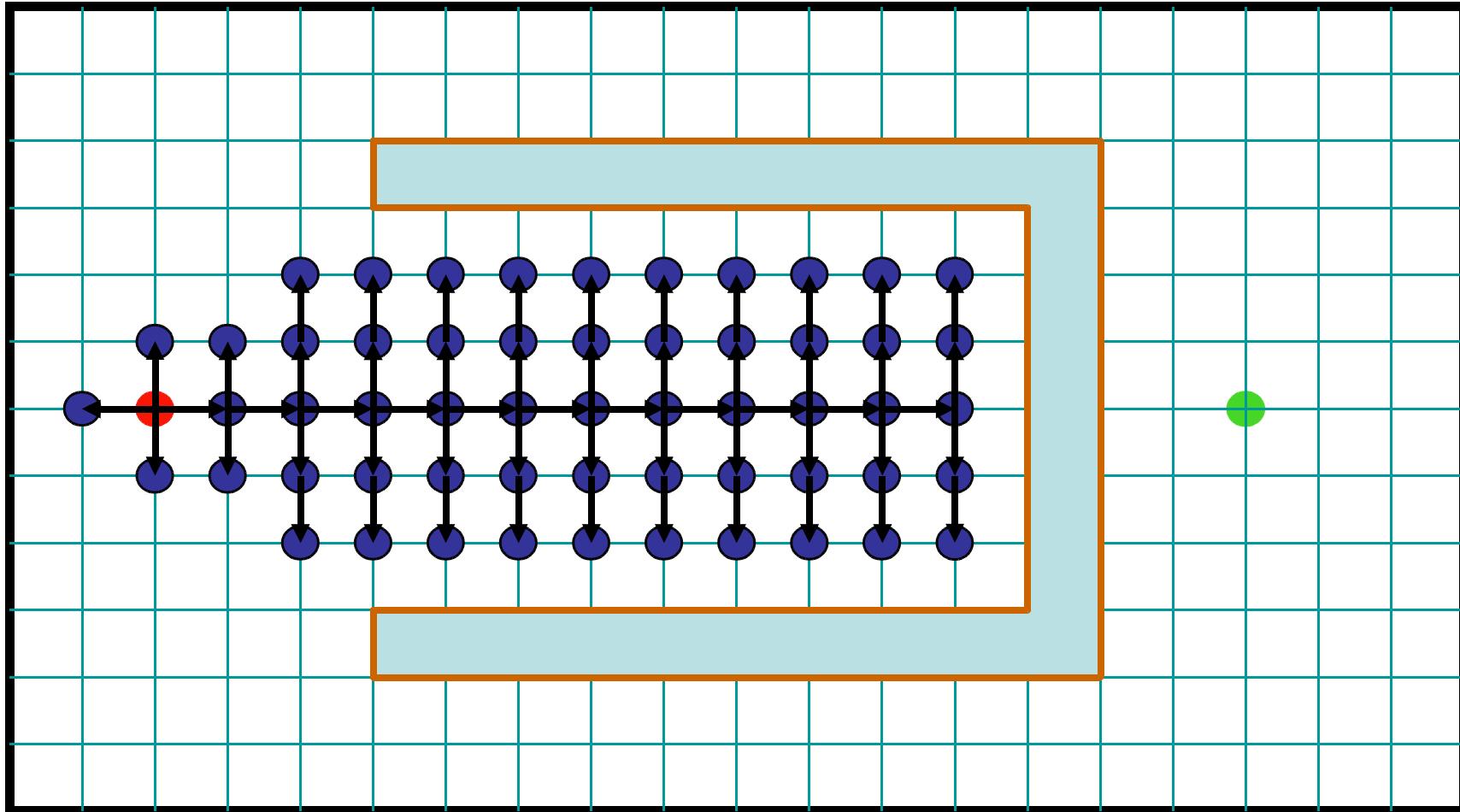
Estratégia: expandir primeiramente um nó com estimativa menos custosa

*Fronteira é uma Fila de prioridade
(com prioridade = $h(x)$)*

Greedy Search

- Expande o nó que parece representar o estado mais próximo do objetivo





A* Search

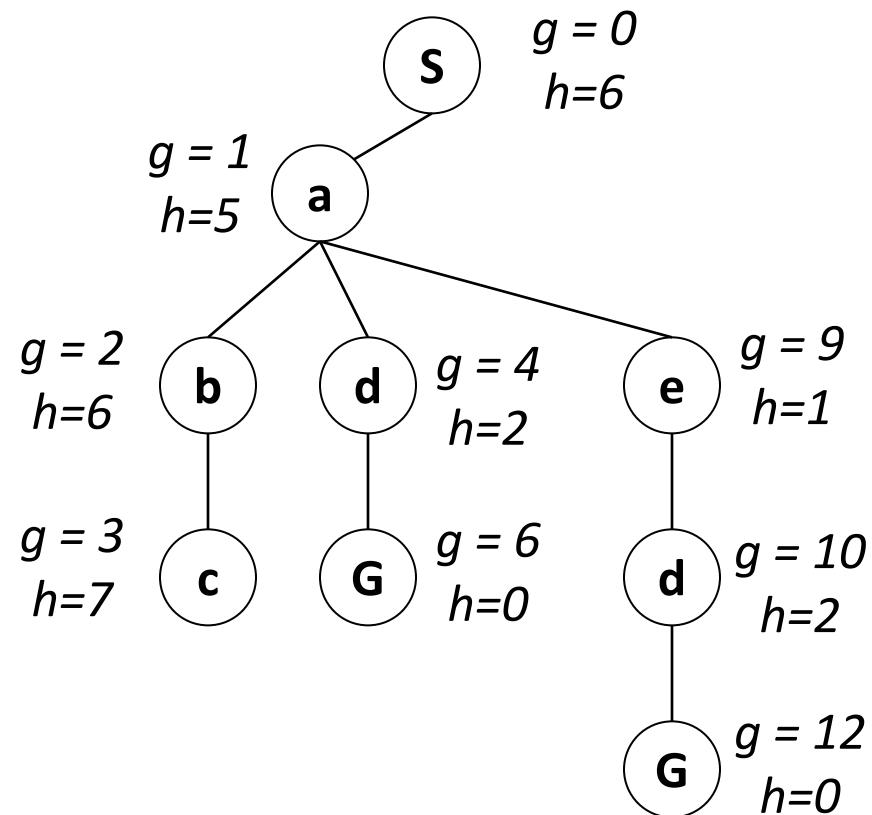
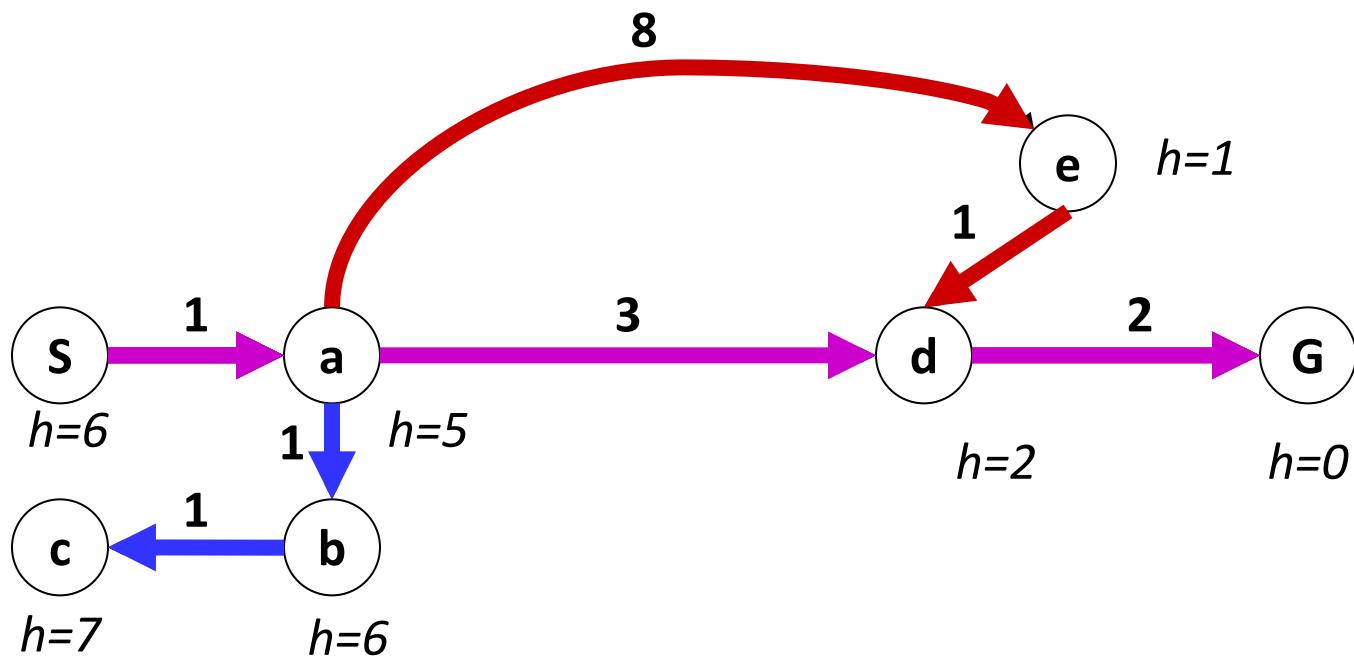


Estratégia: expandir primeiramente um nó cuja estimativa somada ao custo real seja menor

*Fronteira é uma Fila de prioridade
(prioridade = $h(x)$ + custo real)*

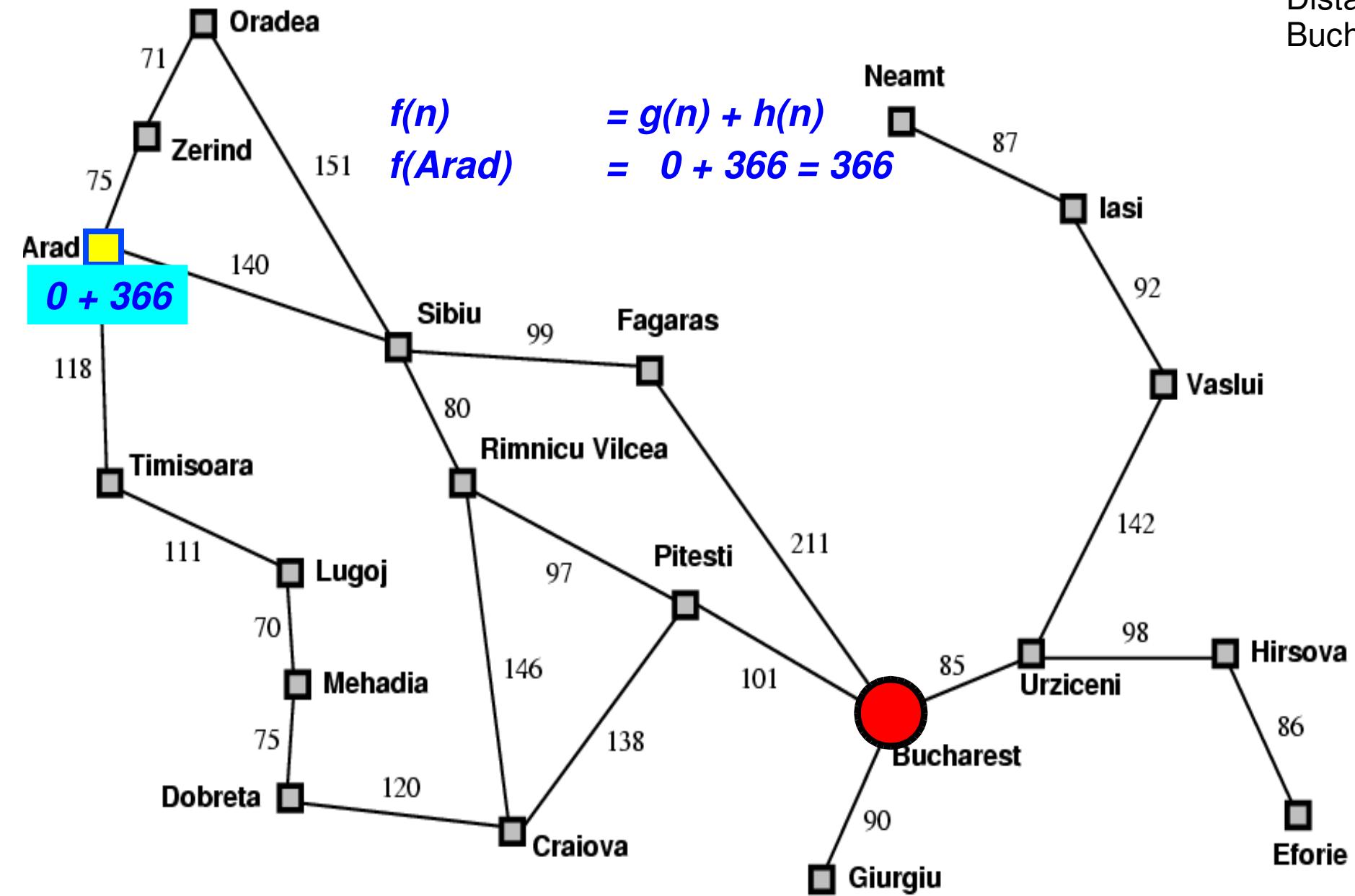
A* = UCS + Greedy

- **Uniform-cost** ordena por custo do caminho: *backward cost*: $g(n)$
- **Greedy** ordena por proximidade do objetivo: *forward cost*: $h(n)$



- **A* Search** ordena pela soma: $f(n) = g(n) + h(n)$

Distância em linha reta para Bucharest:



$$\begin{aligned}
 f(n) &= g(n) + h(n) \\
 f(\text{Arad}) &= 0 + 366 = 366
 \end{aligned}$$

Neamt

87

Iasi

92

Vaslui

142

Hirsova

86

Eforie

Giurgiu

90

Bucharest

101

85

Urziceni

98

Pitesti

138

97

Fagaras

99

Rimnicu Vilcea

80

Sibiu

118

$f(n)$

$f(\text{Arad})$

75

71

Oradea

111

70

Lugoj

75

Mehadia

120

Dobreta

Timisoara

118

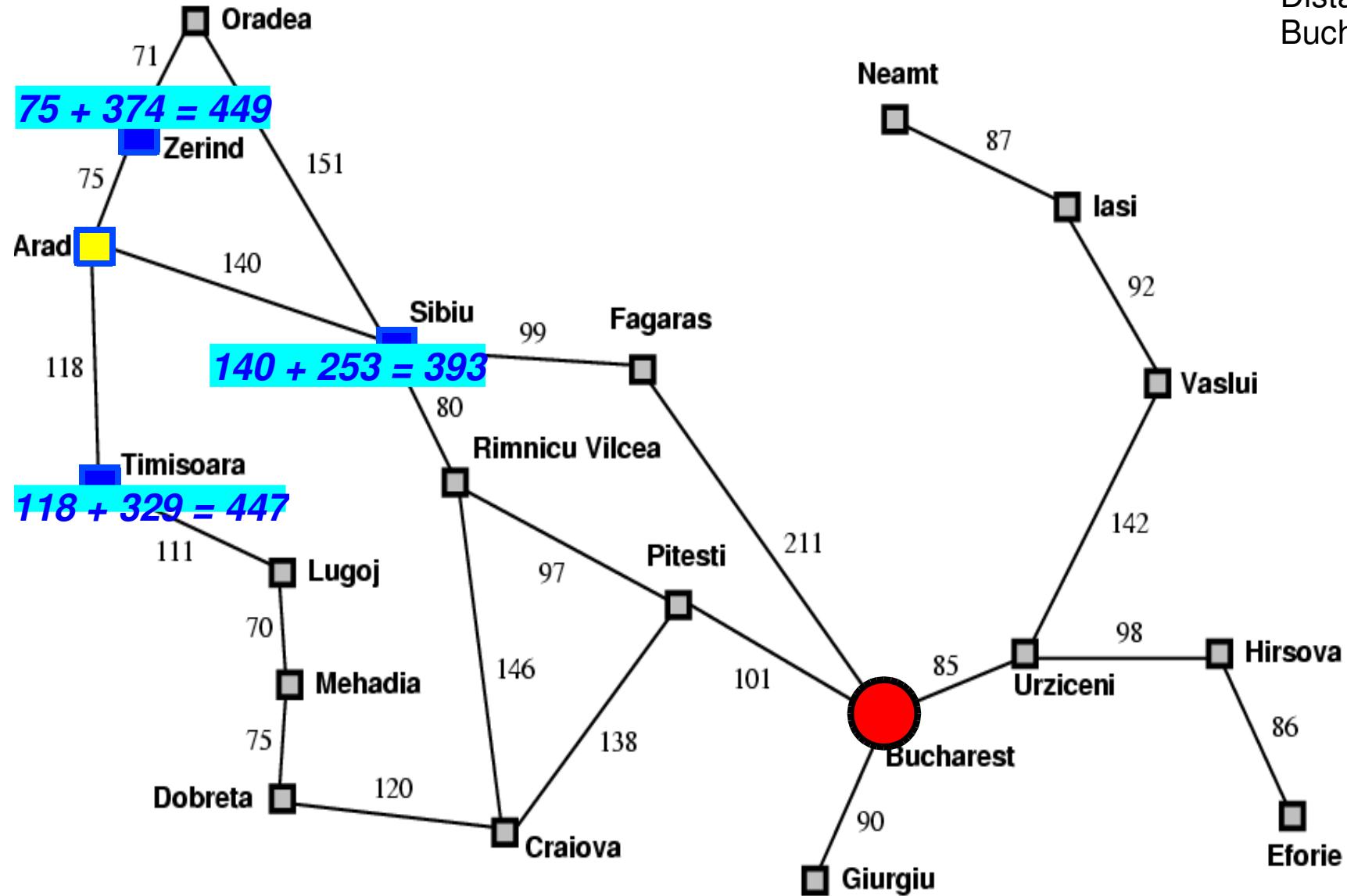
140

151

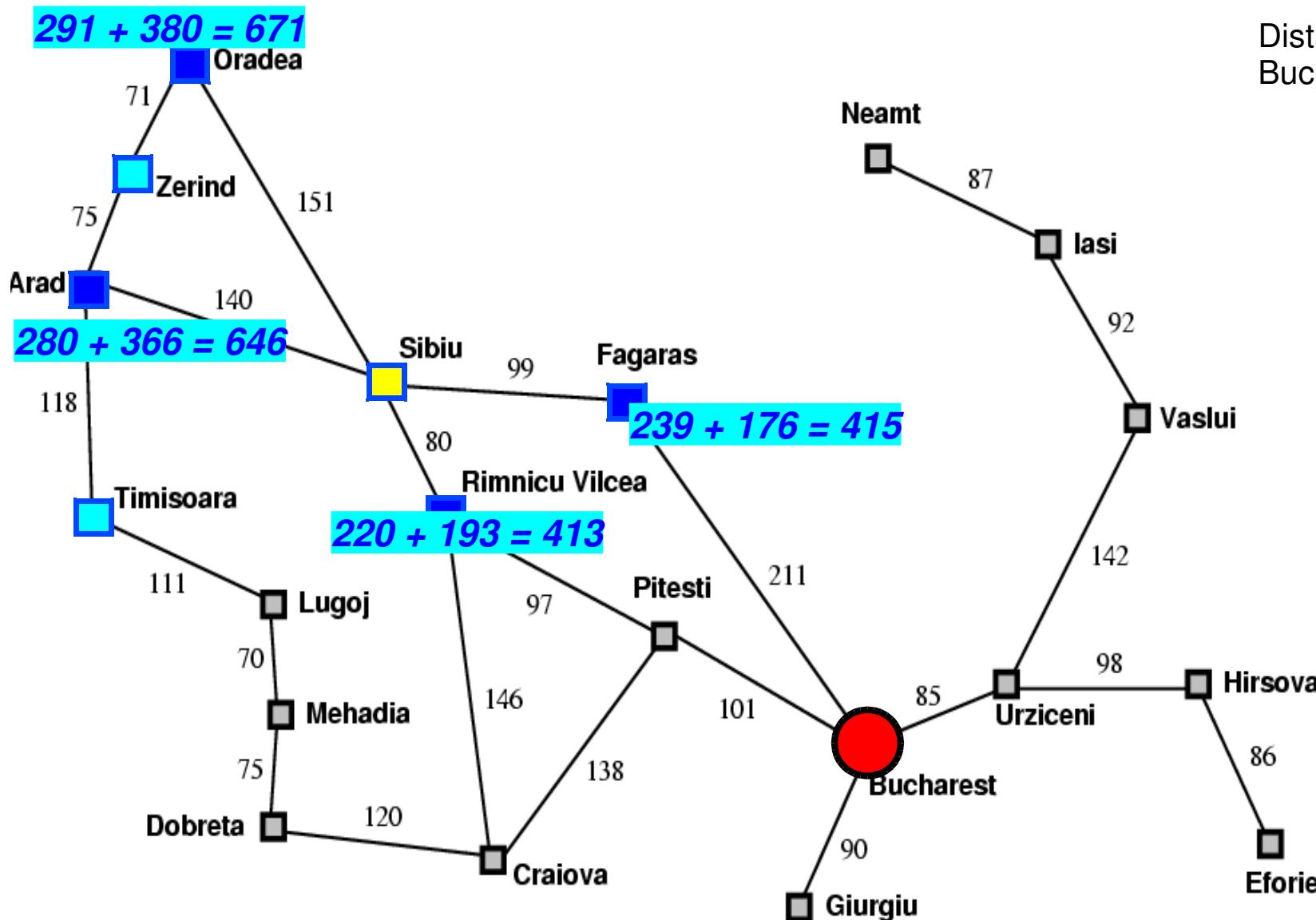
Zerind

$0 + 366$

{Arad₃₆₆}
 {Sibiu₃₉₃, Timi₄₄₇, Zerind₄₄₉}



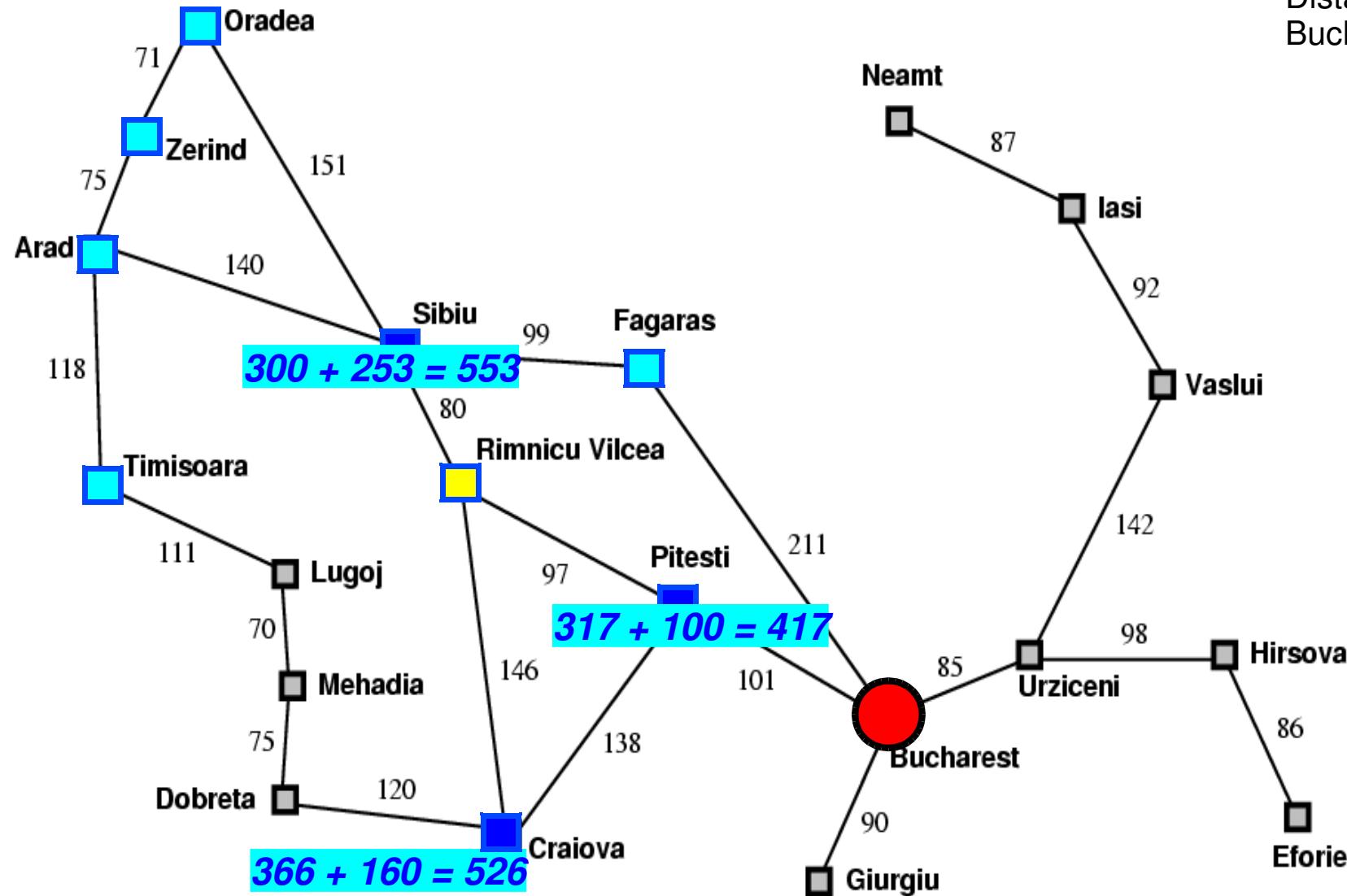
$\{Arad_{366}\}$
 $\{Sibiu_{393}, Timi_{447}, Zerind_{449}\}$
 $\{Rimni_{413}, Faga_{415}, Timi_{447}, Zerind_{449}, Arad_{646}, Oradea_{671}\}$



Distância em linha reta para Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

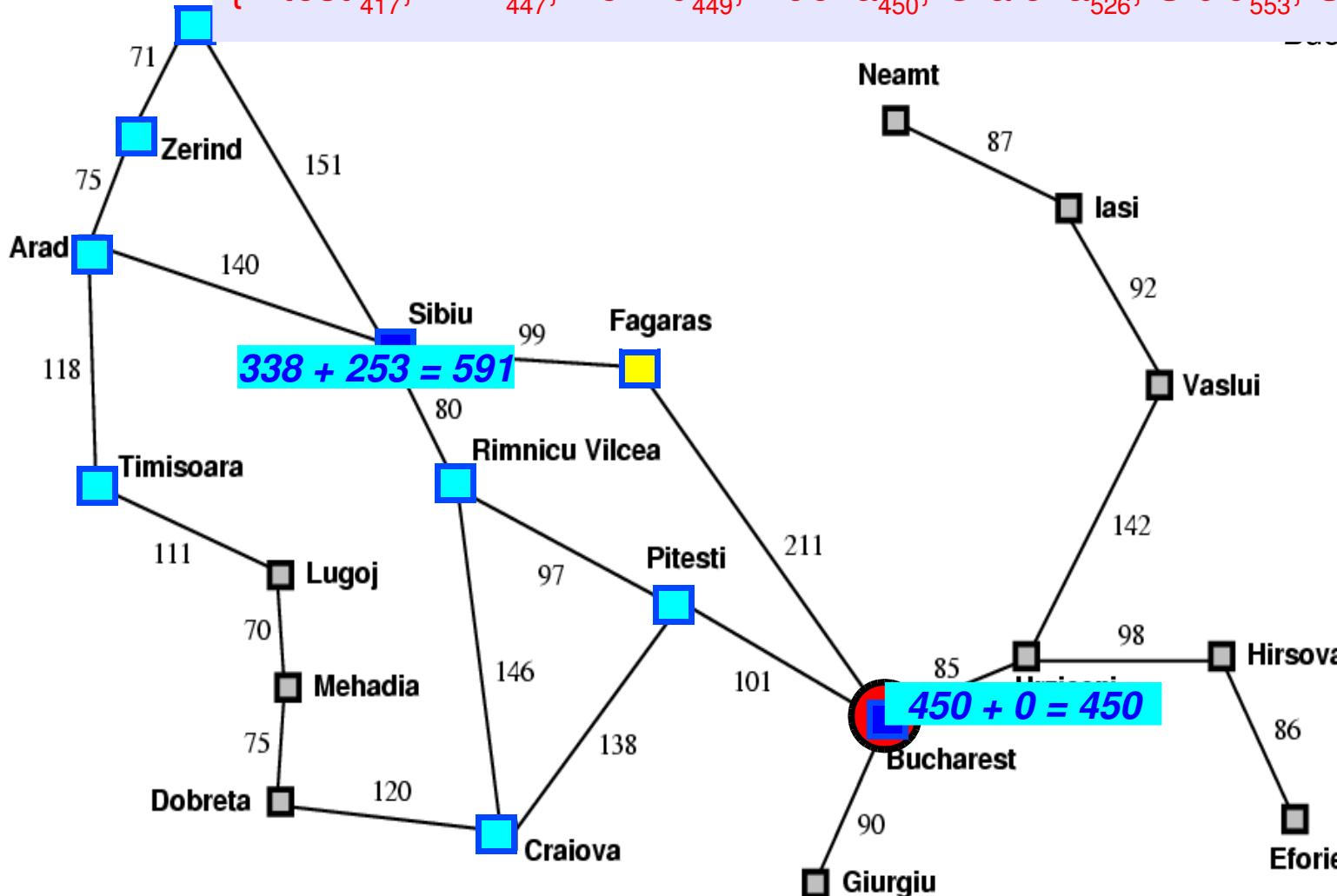
{Arad₃₆₆}
 {Sibiu₃₉₃, Timi₄₄₇, Zerind₄₄₉}
 {Rimni₄₁₃, Faga₄₁₅, Timi₄₄₇, Zerind₄₄₉, Arad₆₄₆, Oradea₆₇₁}
 {Faga₄₁₅, Pitesti₄₁₇, Timi₄₄₇, Zerind₄₄₉, Craiova₅₂₆, Sibiu₅₅₃, Arad₆₄₆, Oradea₆₇₁}



Distância em linha reta para
Bucharest:

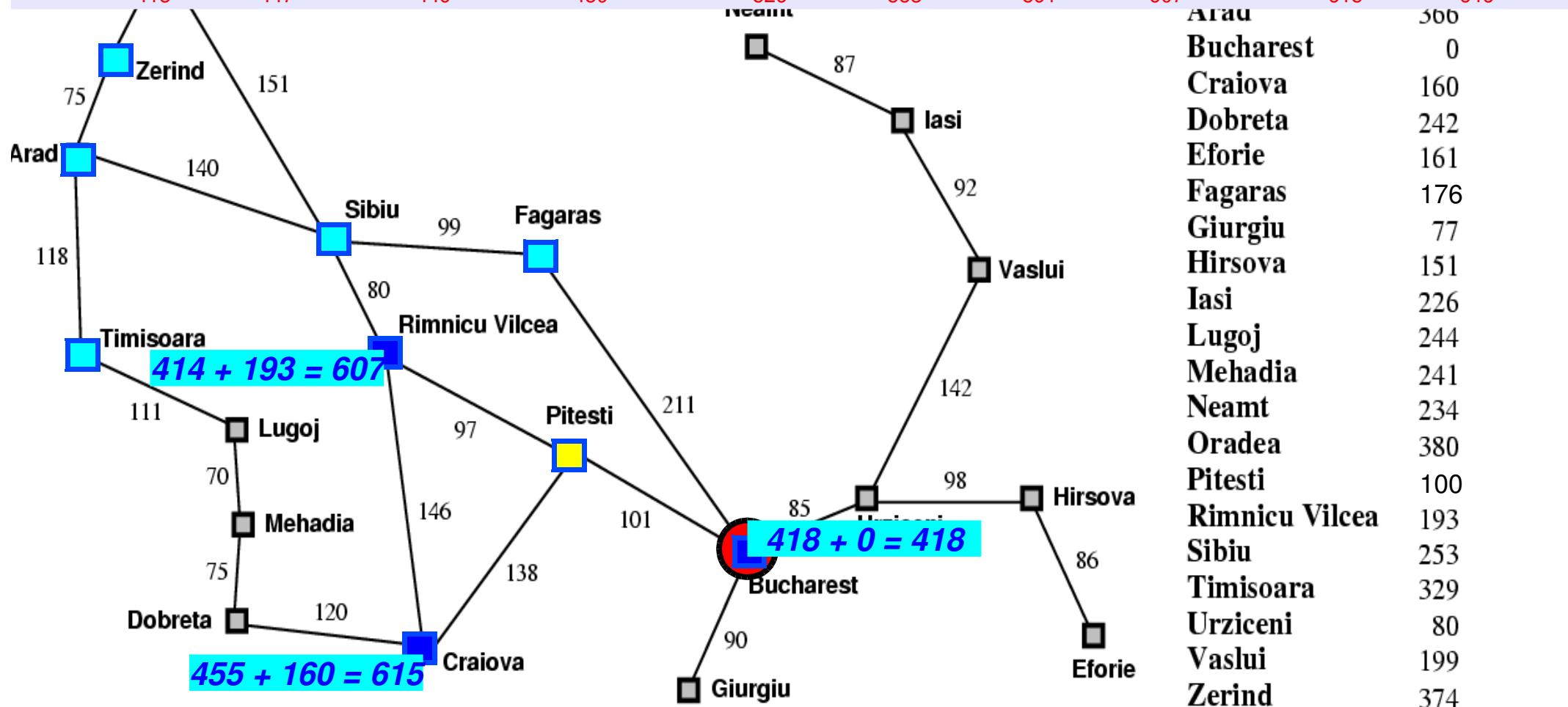
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

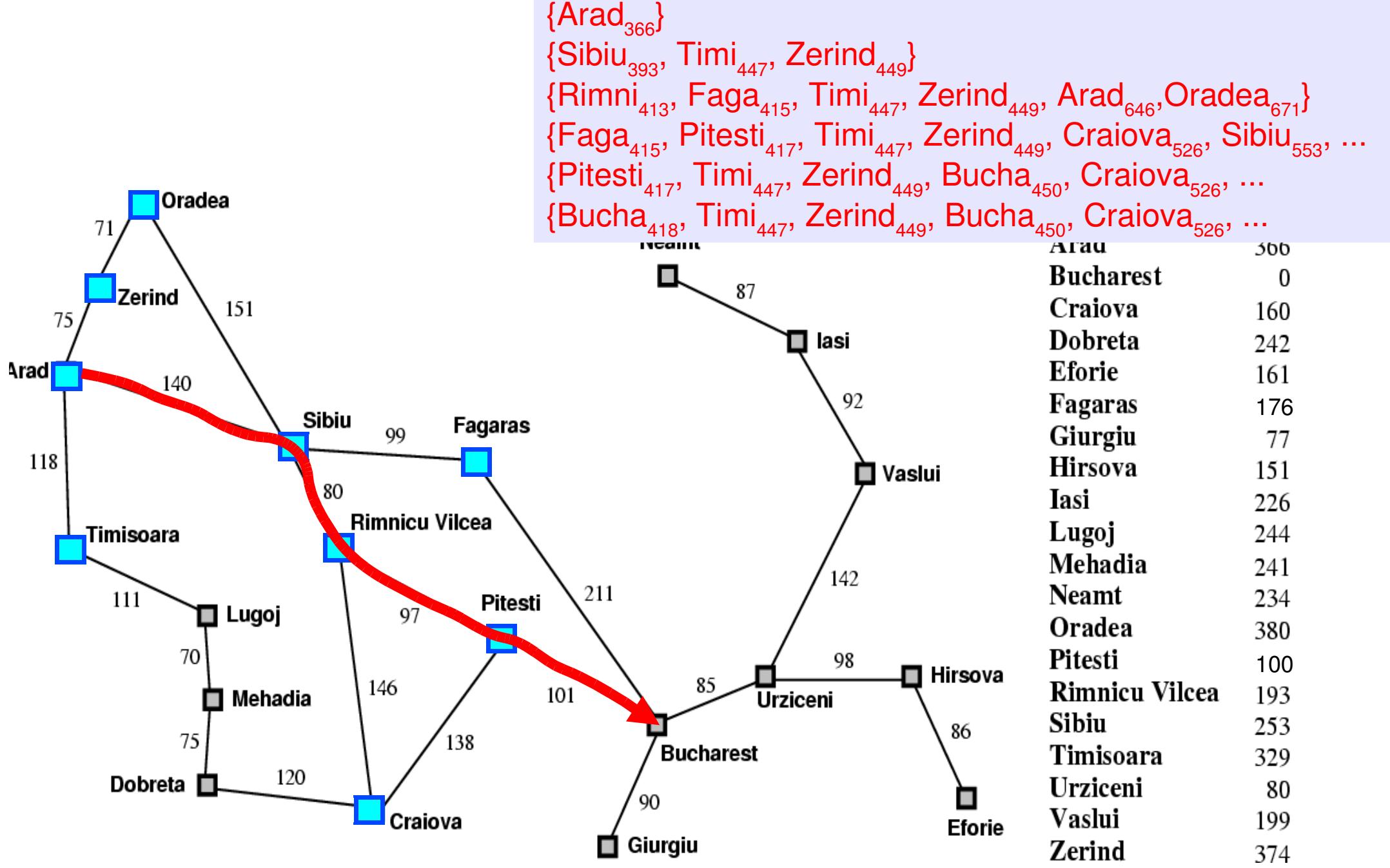
{Arad₃₆₆}
 {Sibiu₃₉₃, Timi₄₄₇, Zerind₄₄₉}
 {Rimni₄₁₃, Faga₄₁₅, Timi₄₄₇, Zerind₄₄₉, Arad₆₄₆, Oradea₆₇₁}
 {Faga₄₁₅, Pitesti₄₁₇, Timi₄₄₇, Zerind₄₄₉, Craiova₅₂₆, Sibiu₅₅₃, Arad₆₄₆, Oradea₆₇₁}
 {Pitesti₄₁₇, Timi₄₄₇, Zerind₄₄₉, Bucha₄₅₀, Craiova₅₂₆, Sibiu₅₅₃, Sibiu₅₉₁, Arad₆₄₆, Oradea₆₇₁}

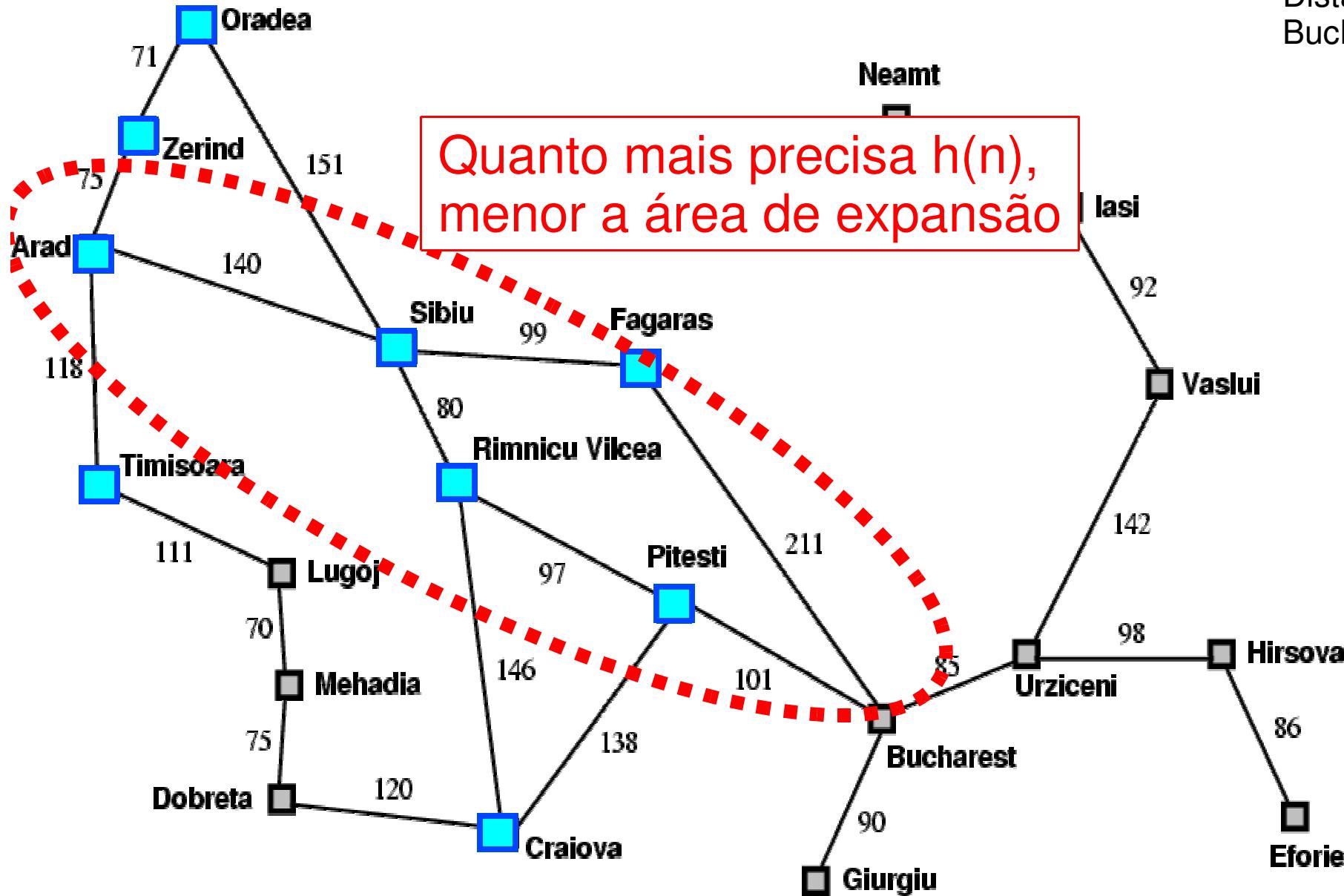


Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

{Arad₃₆₆}
 {Sibiu₃₉₃, Timi₄₄₇, Zerind₄₄₉}
 {Rimni₄₁₃, Faga₄₁₅, Timi₄₄₇, Zerind₄₄₉, Arad₆₄₆, Oradea₆₇₁}
 {Faga₄₁₅, Pitesti₄₁₇, Timi₄₄₇, Zerind₄₄₉, Craiova₅₂₆, Sibiu₅₅₃, Arad₆₄₆, Oradea₆₇₁}
 {Pitesti₄₁₇, Timi₄₄₇, Zerind₄₄₉, Bucha₄₅₀, Craiova₅₂₆, Sibiu₅₅₃, Sibiu₅₉₁, Arad₆₄₆, Oradea₆₇₁}
 {Bucha₄₁₈, Timi₄₄₇, Zerind₄₄₉, Bucha₄₅₀, Craiova₅₂₆, Sibiu₅₅₃, Sibiu₅₉₁, Rimi₆₀₇, Craiova₆₁₅, Arad₆₄₆, Ora





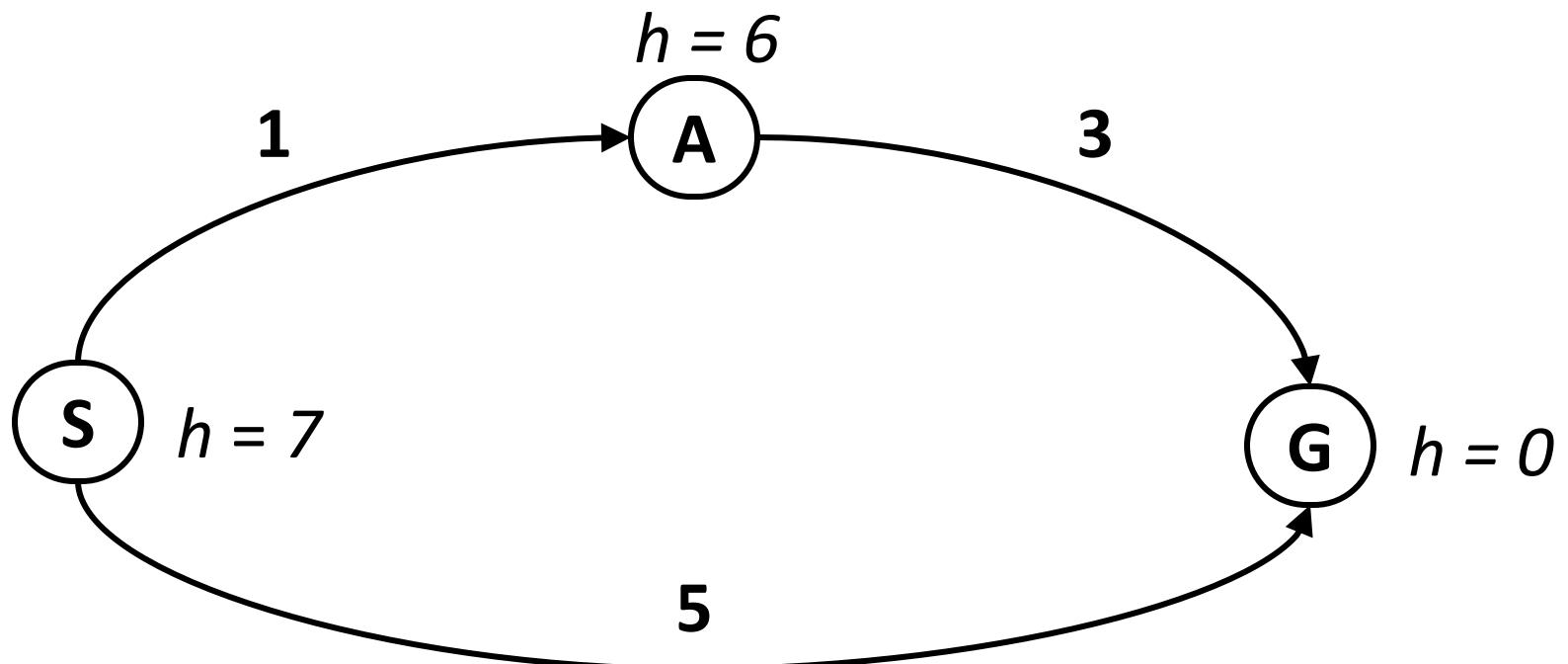


Distância em linha reta para
Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

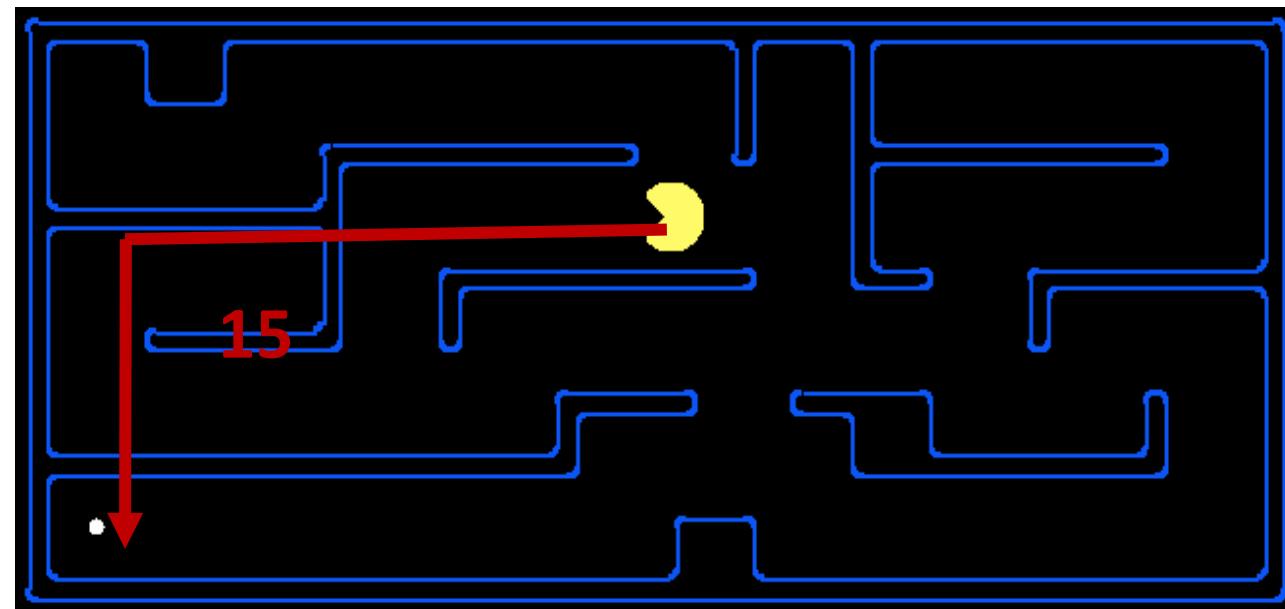
A* : Otimalidade?

A* : Otimalidade?

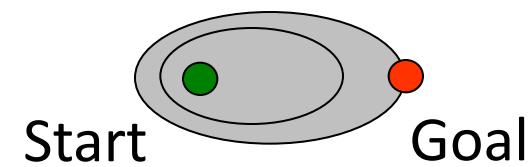
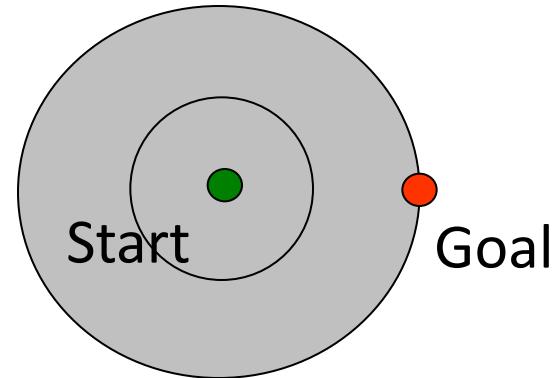
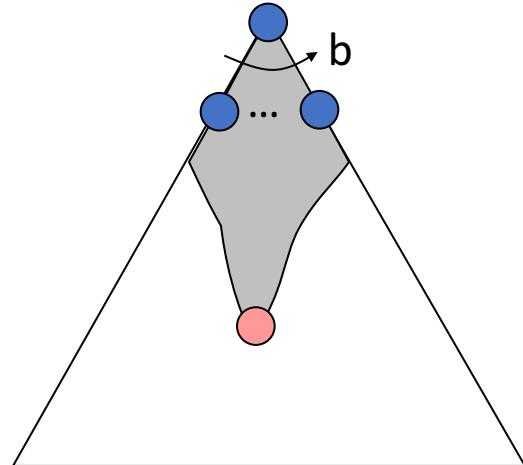
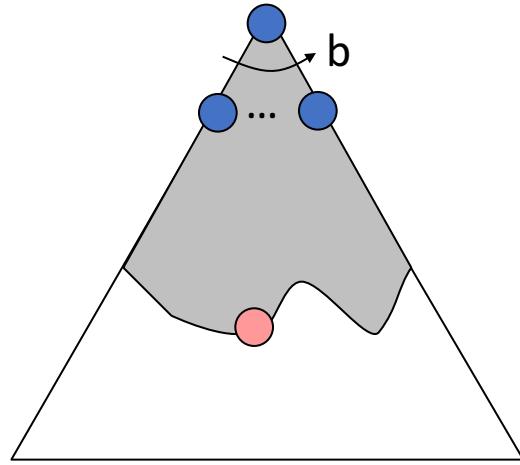


Admissibilidade

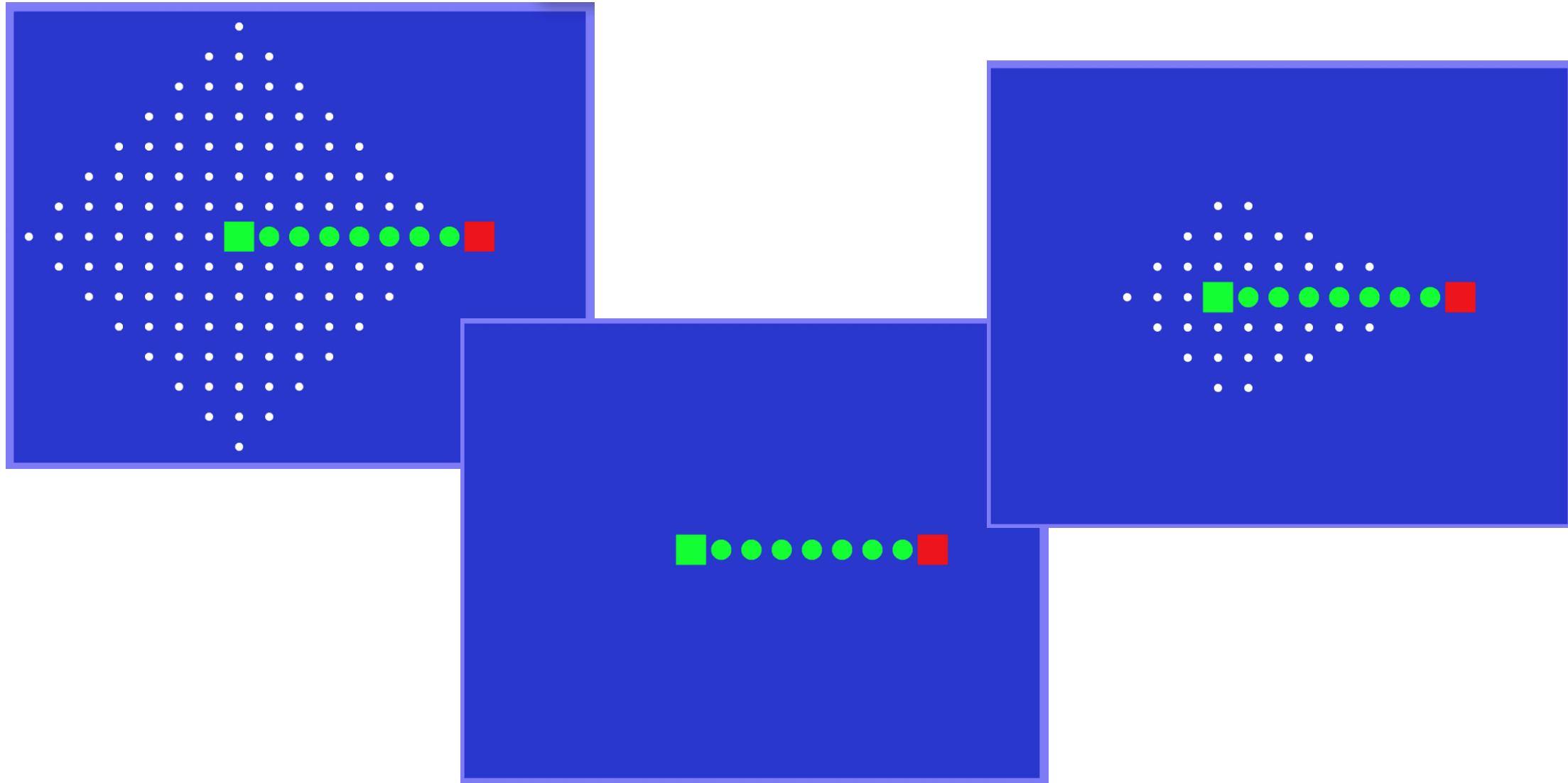
- Heuristica h é *admissível* (otimista) se: $0 \leq h(n) \leq h^*(n)$
onde $h^*(n)$ é o custo real para o estado objetivo mais próximo (não superestima o custo real)



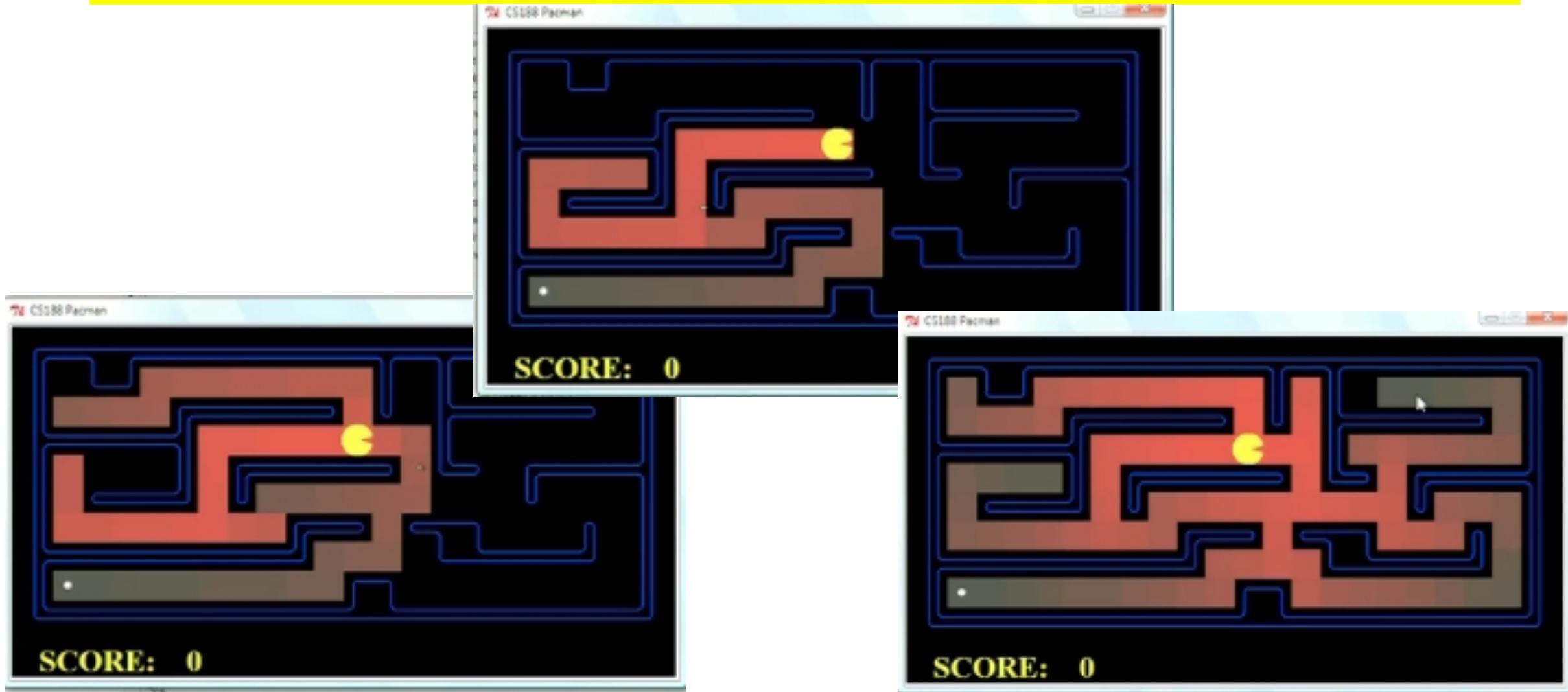
UCS vs A* :: expansão, curvas de contorno



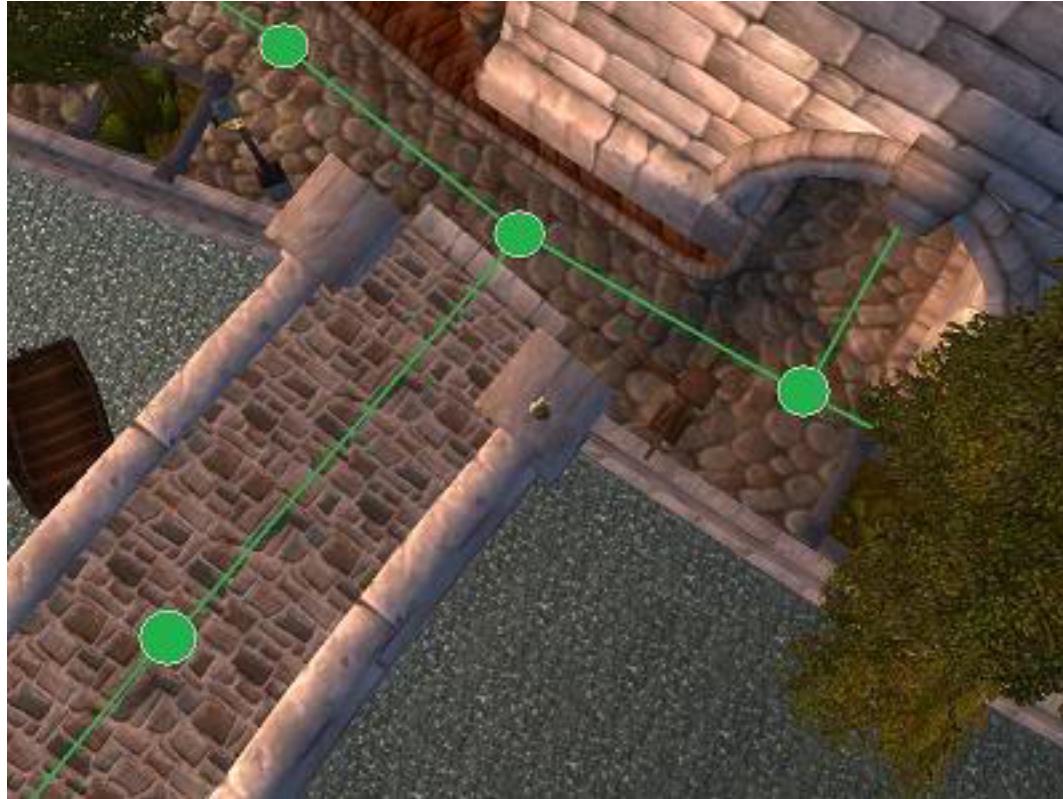
Q: Quem é quem? A*, BFS, UCS



Q: Quem é quem? A*, BFS, UCS



A* para pathfinding em jogos .: World of Warcraft



Navegação dos NPCs

A* para pathfinding em jogos .: World of Warcraft

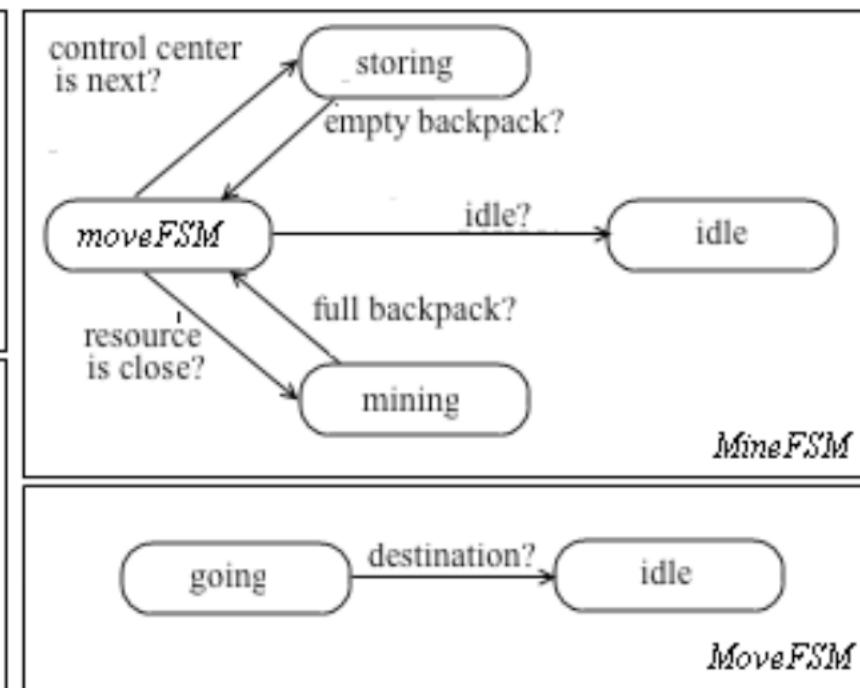
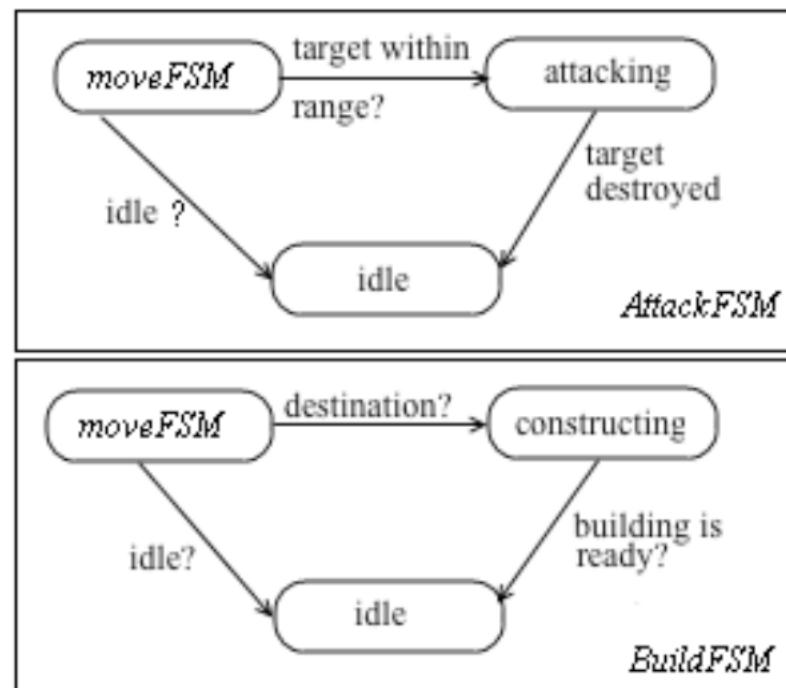
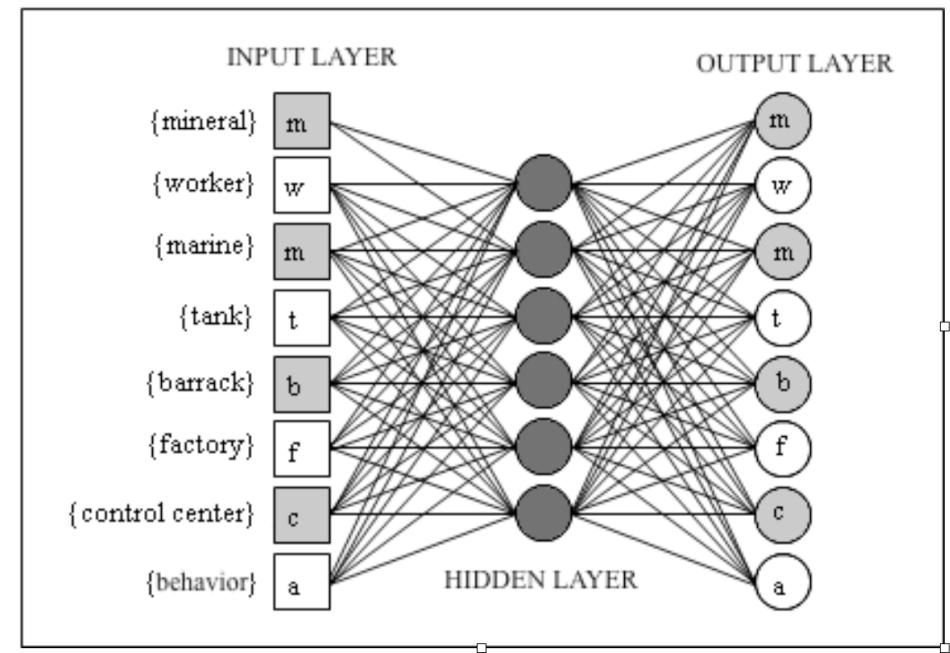
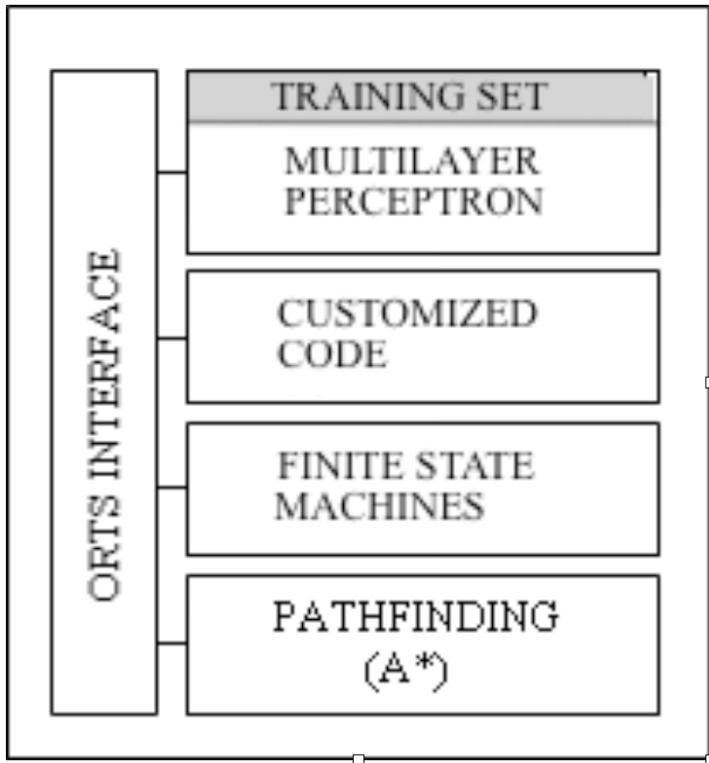


Navegação dos NPCs

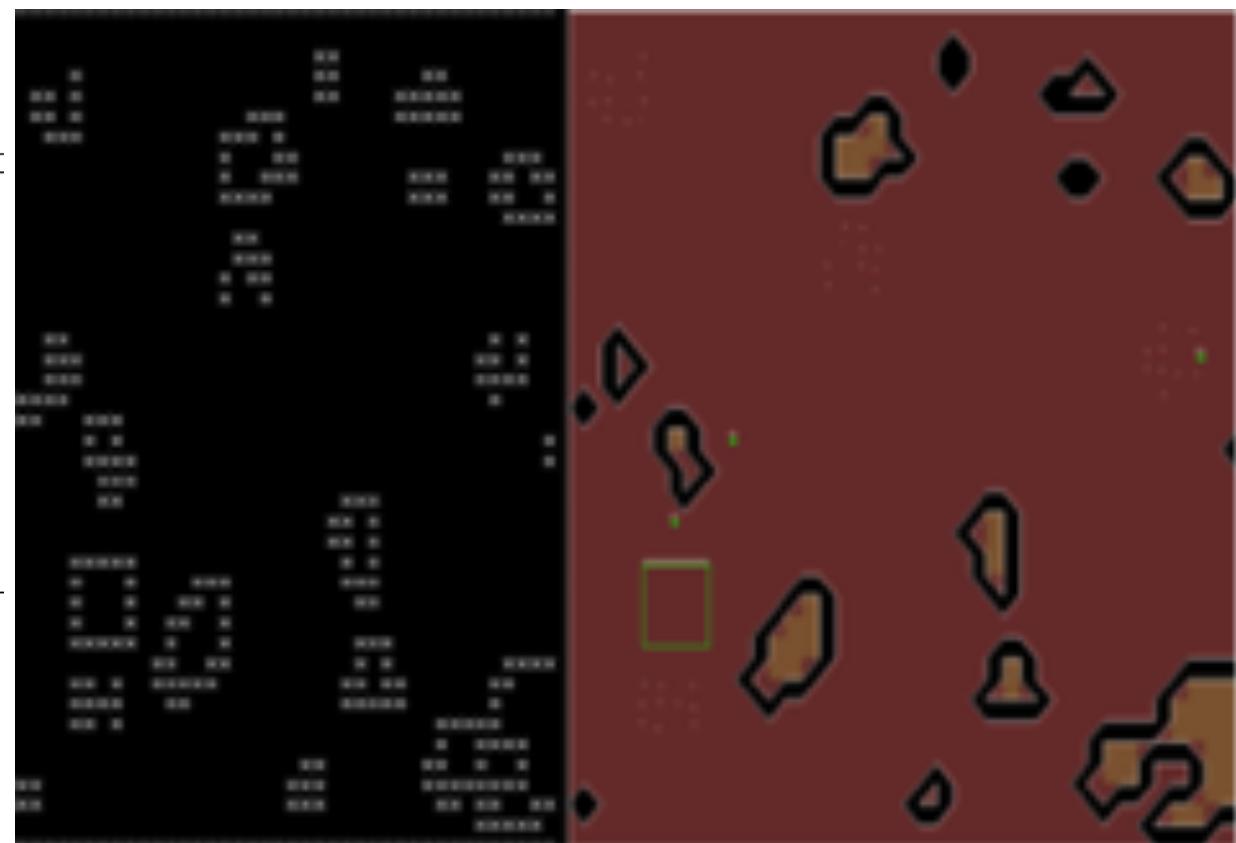
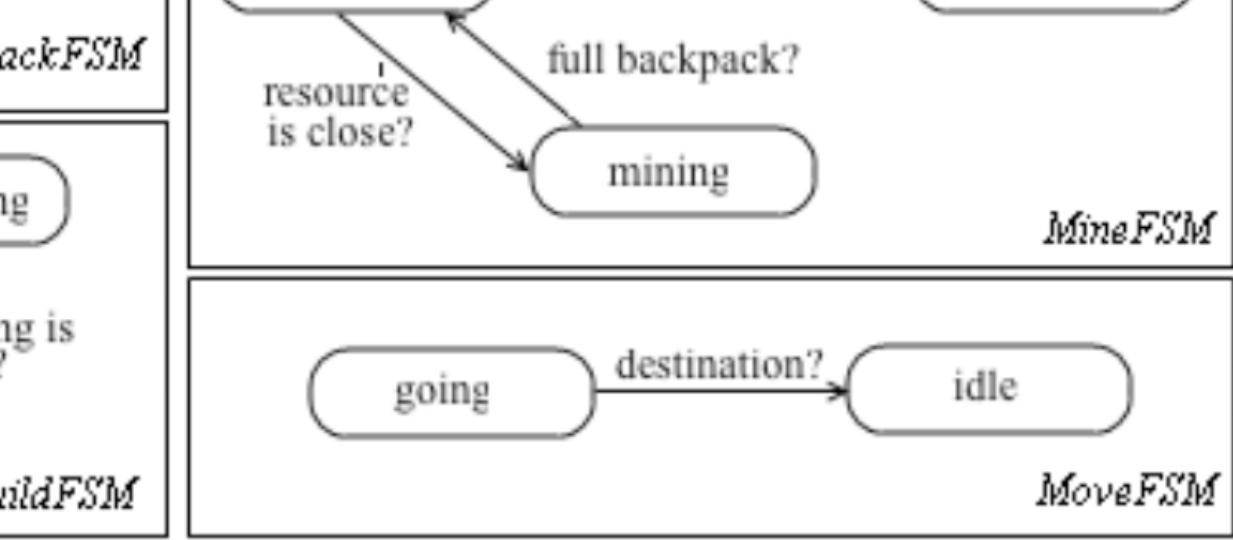
RTS game engine: ORTS (competição anual)



MACEDO, H. T., SILVA, A. S. Personalized Player Character in Real Time Strategy Games. European Journal of Scientific Research, v. 111, p. 530-551, n. 2013.



MACEDO, H. T., SILVA, A. S. Personalized Player Character
in Real Time Strategy Games. European Journal of
Scientific Research, v. 111, p. 530-551, n. 2013.



Como escolher h

h depende de cada problema particular

h deve ser *admissível*

Estratégia geral:

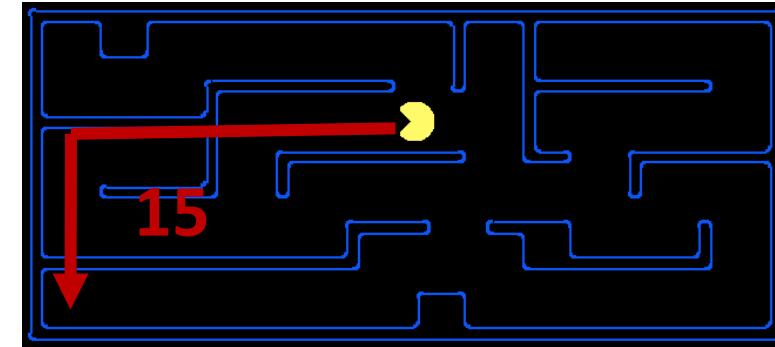
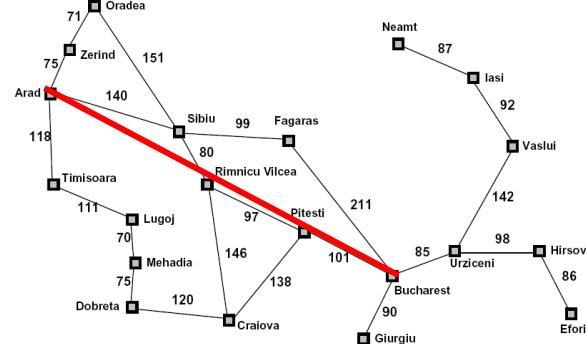
Simplificar restrições do problema



Admissibilidade da heurística

- Maior parte do trabalho em resolver difíceis problemas de busca de forma ótima está em definir uma heurística admissível.
- Frequentemente, heurísticas admissíveis são soluções para *relaxed problems* (*simplificações das restrições do problema original*)

366



2	8	3
1	6	4
7		5



1	2	3
8		4
7	6	5

Início

Objetivo

Restrição real: Um número pode mover-se de A para B se A é adjacente a B e B está vazio

Busca cega:

solução média em 20 passos
fator de ramificação médio: 3
 ≈ 320 estados possíveis

2	8	3
1	6	4
7		5



1	2	3
8		4
7	6	5

Inicio

Objetivo

Removendo restrições:

h1 : Uma peça pode mover p/ qualquer lugar

h2 : Uma peça pode mover p/ um lugar adjacente

2	8	3
1	6	4
7		5

h1 = 4



1	2	3
8		4
7	6	5

$$\textbf{h2} = 1 + 1 + 0 + 0 + 0 + 1 + 0 + 2 = 6$$

Início

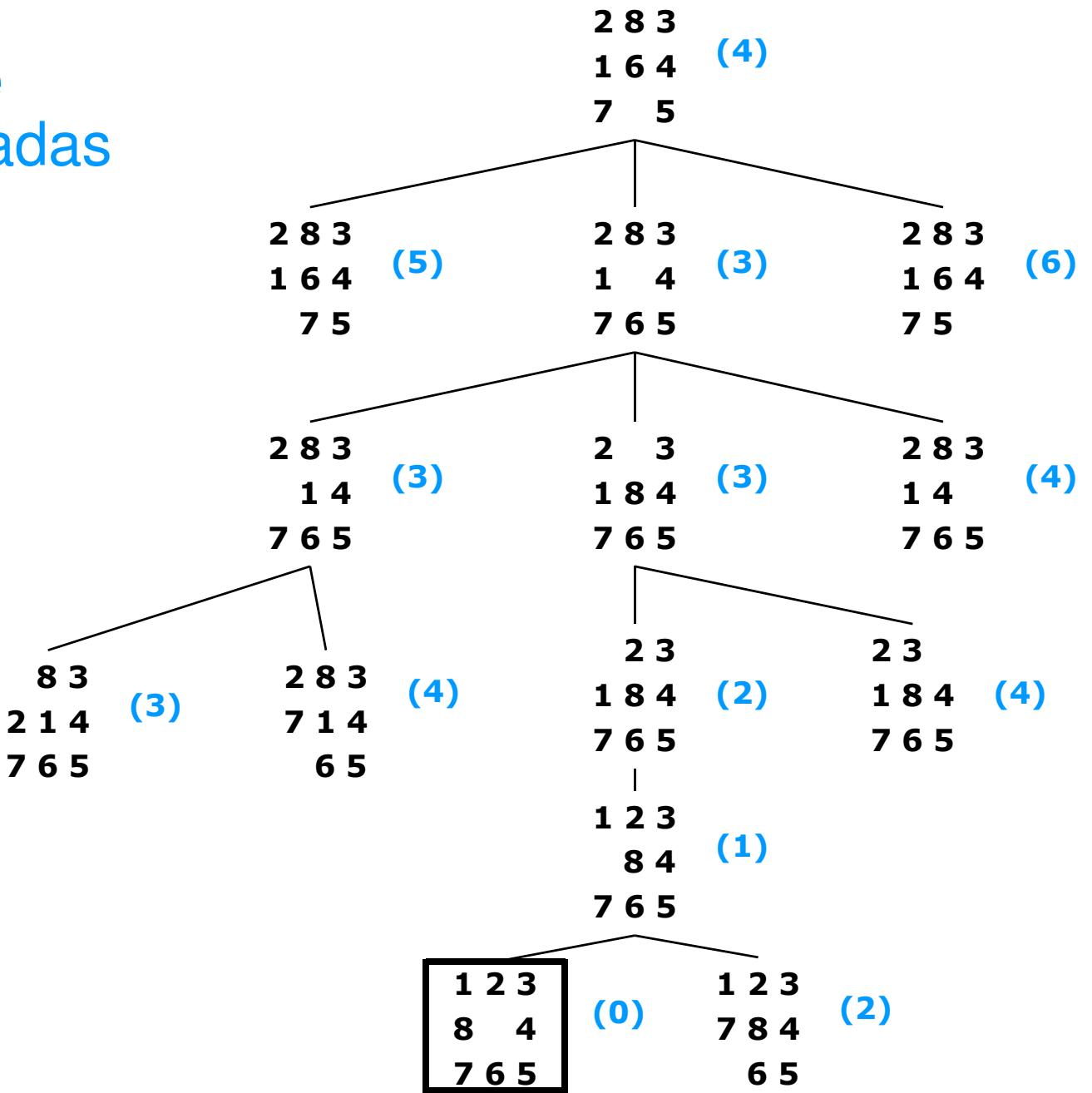
Objetivo

Algumas heurísticas possíveis:

h1 = n0 de elementos em posições erradas

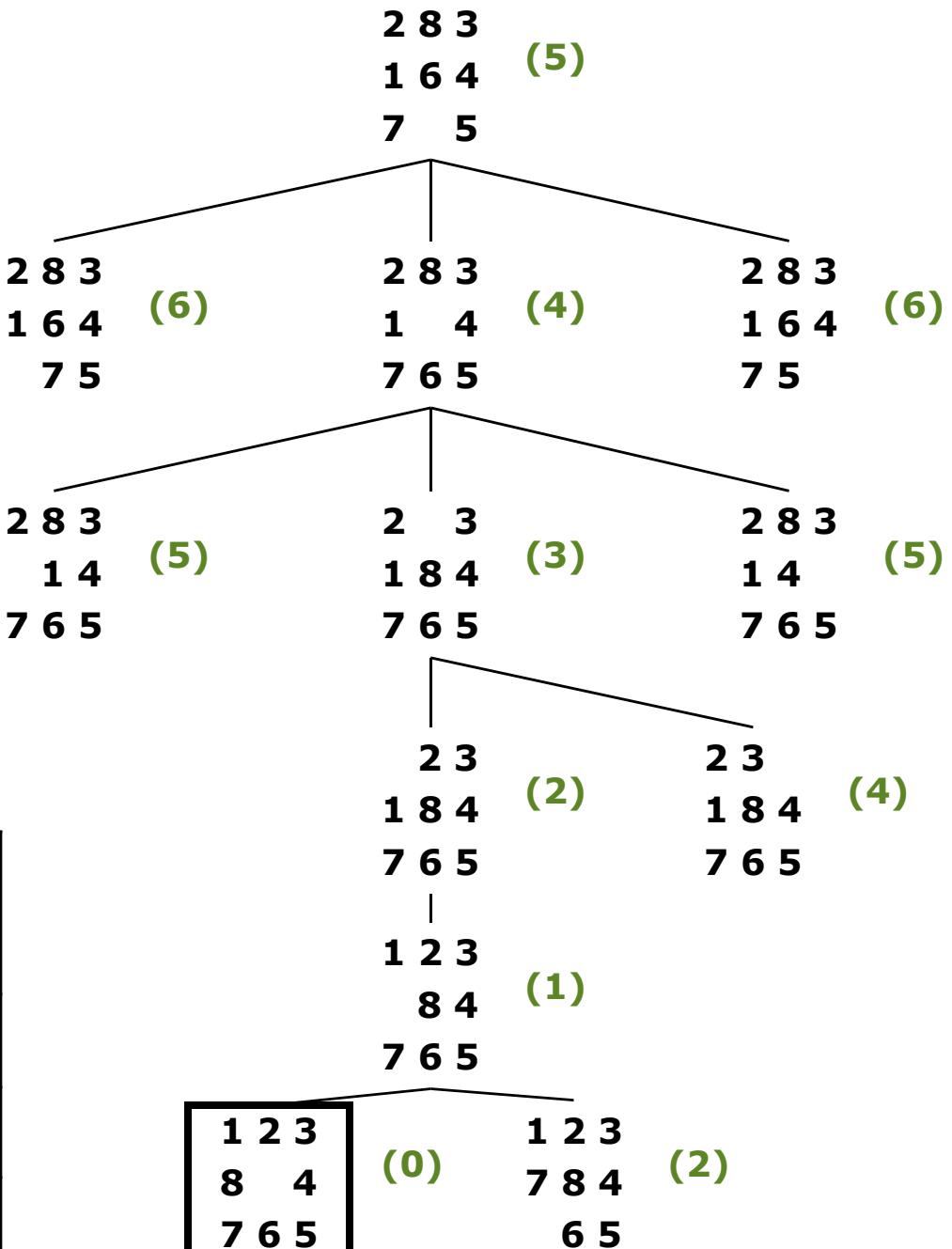
h2 = soma das distâncias de cada elemento à posição objetivo
(Manhattan distance => |x-u| + |y-v|)

Considerando $h_1 = \text{n}_0$ de elementos em posições erradas



Considerando $h2$ = soma das distâncias

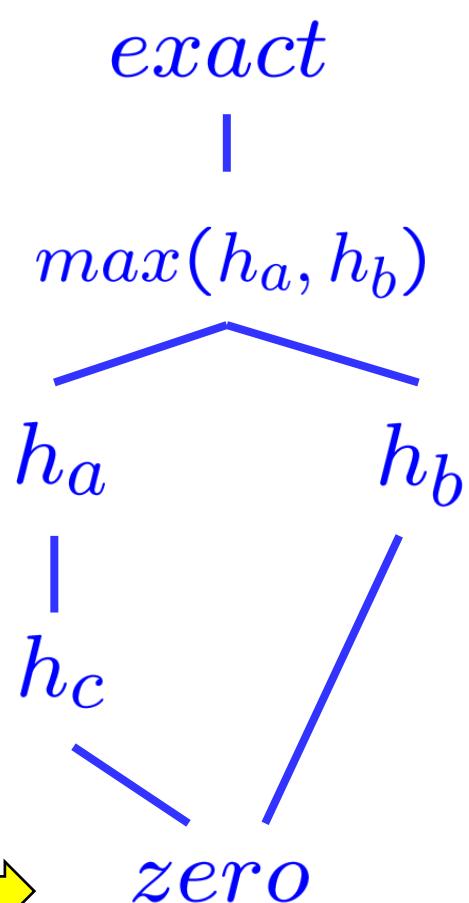
	Average nodes expanded when the optimal path has...		
	...4 steps	...8 steps	...12 steps
$h1$	13	39	227
$h2$	12	25	73



Qualidade da heurística: Dominância

- h_a domina h_c se $\forall n : h_a(n) \geq h_c(n)$

$$h(n) = \max(h_a(n), h_b(n))$$



Q: O que significa?

Qualidade de h



Medida através do **fator de expansão efetivo (b^*)**.

Árvore com nível de profundidade d

$N+1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$, onde
 N = total de nós expandidos para uma instância de problema
 d = profundidade da solução;

Mede-se empiricamente a qualidade de h a partir do conjunto de valores experimentais de N e d .

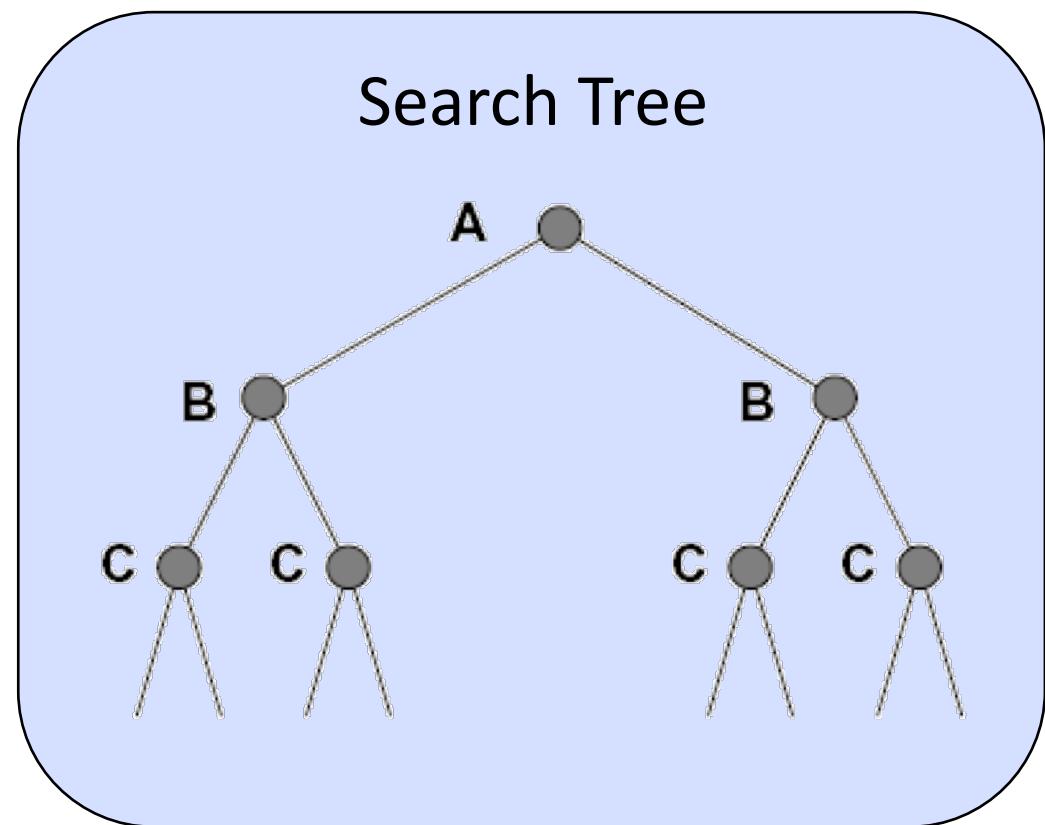
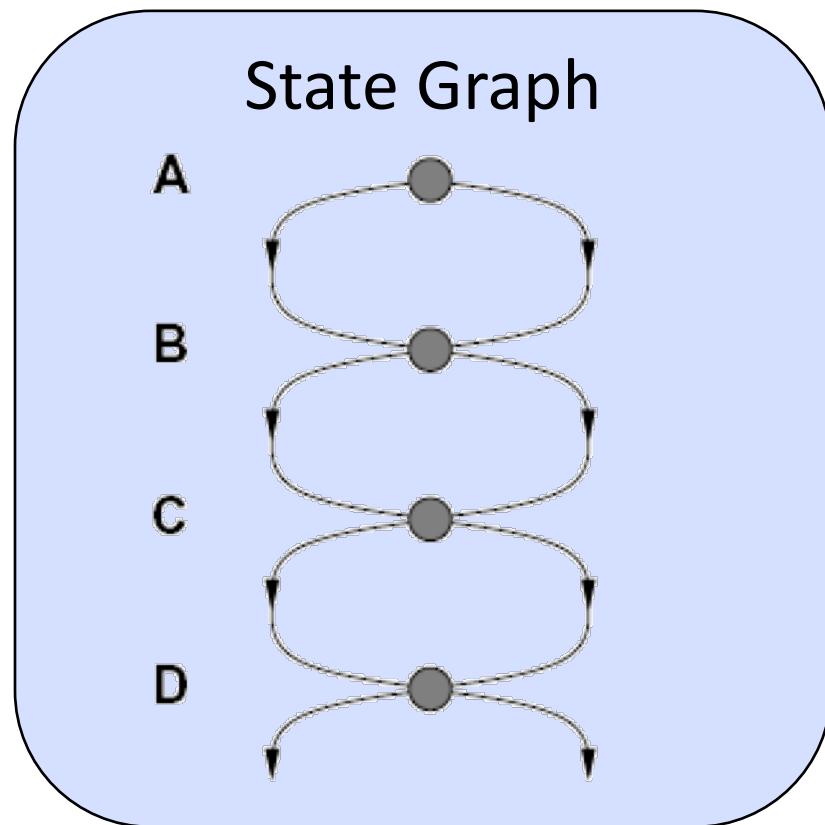
→ h será tão melhor quanto mais próximo de 1, b^* for.

Experimento: 1200 execuções para o problema 8-números

d	Custo da busca			b^*		
	BPI	$A^*(h_1)$	$A^*(h_2)$	BPI	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
8	6384	39	25	2.80	1.33	1.24
12	364404	227	73	2.78	1.42	1.24
16	–	1301	211	–	1.45	1.25
20	–	7276	676	–	1.47	1.27
24	–	39135	1641	–	1.48	1.26

Busca em árvore: questões a se considerar!

- Falha em detectar estados repetidos pode causar trabalho extra exponencialmente.



Busca em árvore: questões a se considerar!

- Nunca **expandir** o mesmo estado de novo
- Como implementar:
 - Tree search + conjunto de estados (**closed set**)
 - Antes de expandir um nó, verificar se este não já foi expandido antes
 - Se já foi, ignore-o; se não foi, adiciona ao closed set

=> Busca em grafo

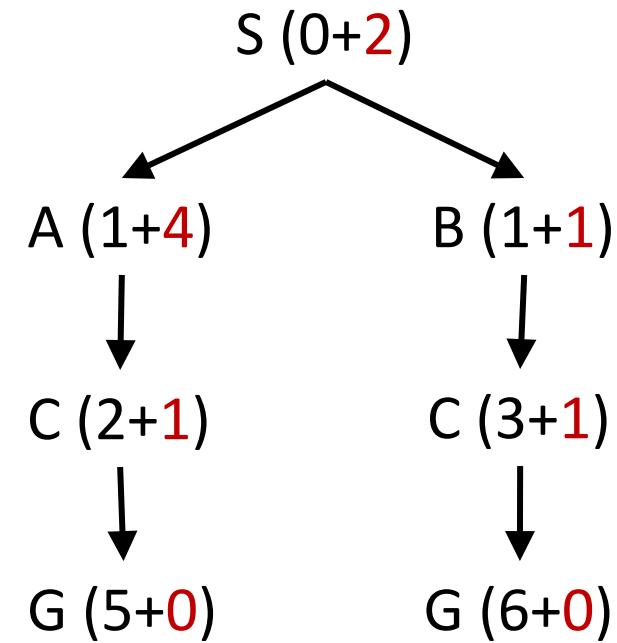
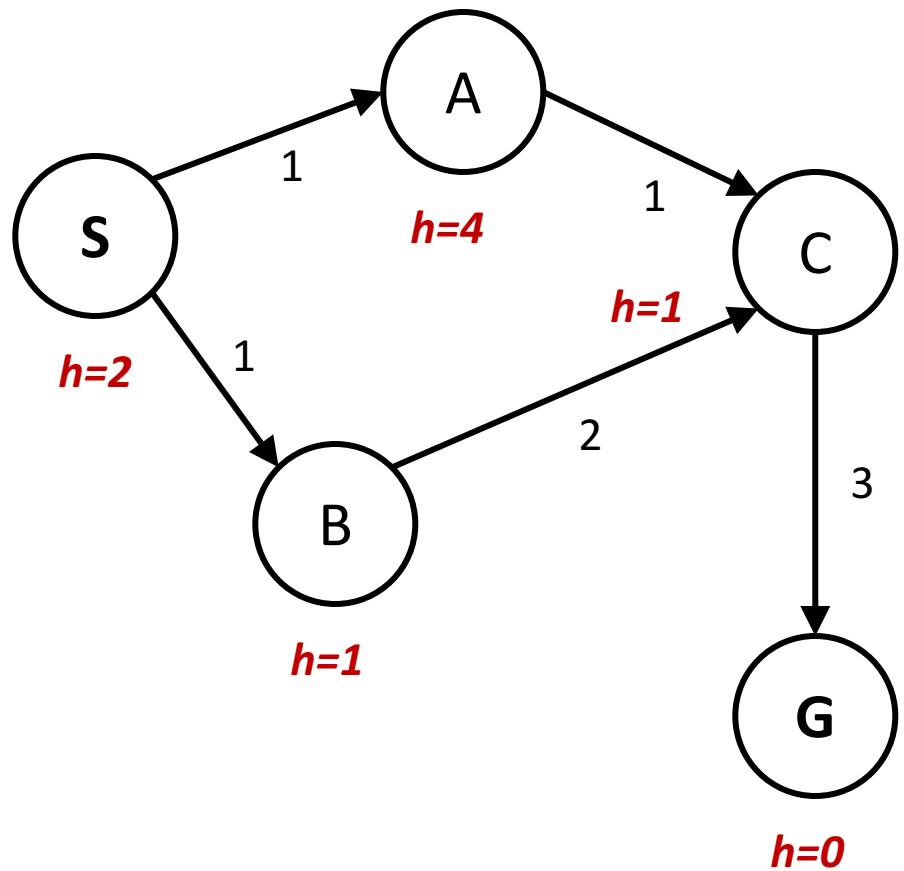
Prejudica Completude do A*? Não

Prejudica a otimalidade do A*? Sim

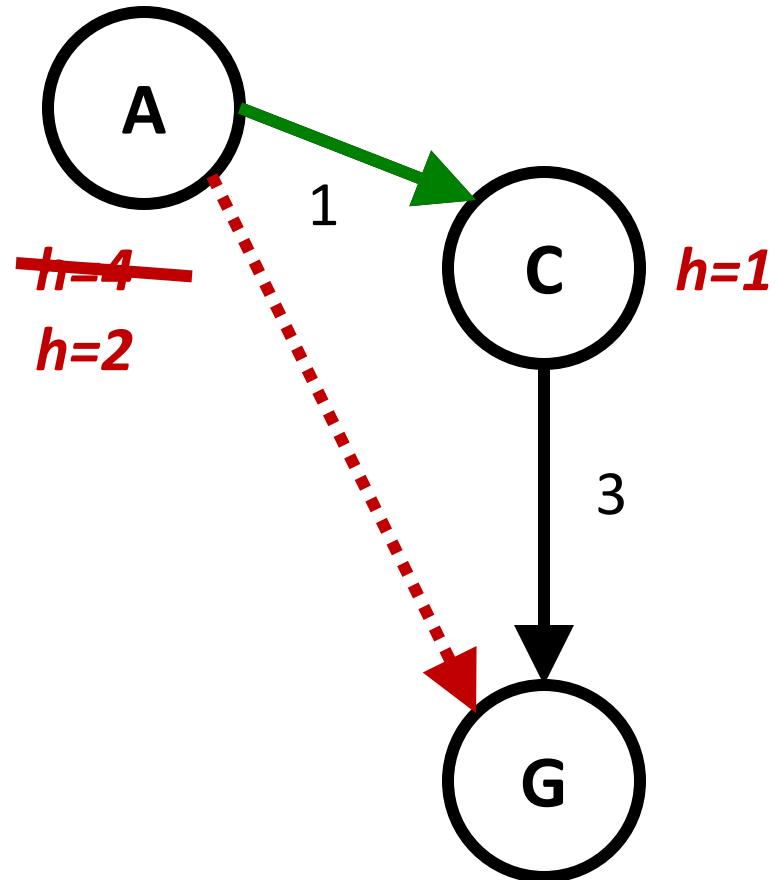
Busca em grafo

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
  closed ← an empty set
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    if STATE[node] is not in closed then
      add STATE[node] to closed
      for child-node in EXPAND(STATE[node], problem) do
        fringe ← INSERT(child-node, fringe)
    end
  end
```

Q: Identificam o problema?

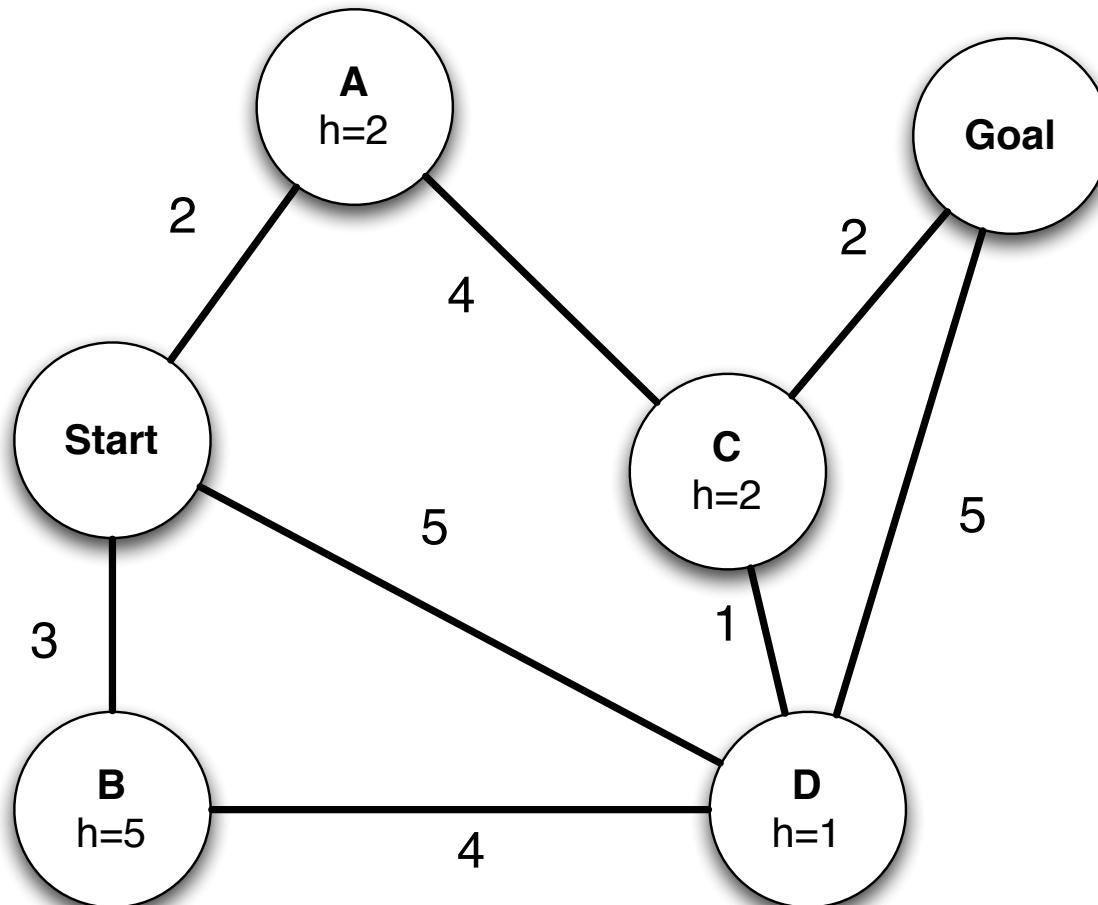


Consistência da heurística



- Admissibilidade: heurística \leq custo real p/ objetivo
 $h(A) \leq$ custo real de A para G
- Consistência: heurística \leq custo real p/ cada estado
 $h(A) - h(C) \leq$ custo(A para C)
- Consequências:
 - A função f ao longo do caminho para o objetivo nunca diminui
 $h(A) \leq$ custo(A para C) + $h(C)$
 - Busca em grafo A* é ótima!!!

Q: Para cada estratégia de busca em grafo, mostre a ordem em que os estados são expandidos e o caminho da solução retornado pela busca (empates resolvem-se em ordem alfabética)



a) Uniform-Cost Search

Estados expandidos: Start, A, B, D, C, Goal
Caminho retornado: Start-A-C-Goal

b) Greedy

Estados expandidos: Start, D, Goal
Caminho retornado: Start, D, Goal

c) A*

Estados expandidos: Start, A, D, B, C, Goal
Caminho retornado: Start-A-C-Goal

Resumo

- Busca em árvore:
 - A* é ótima se heurística é admissível
 - UCS é um caso especial onde $h = 0$
- Busca em grafo:
 - A* é ótima se heurística é consistente
 - UCS é ótima porque $h = 0$ é heurística consistente
- Consistência => Admissibilidade,
 - Mas não o contrário!



Hendrik Macedo

Escreve sobre Inteligência Artificial no Saense.

<http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/>